

UNIVERSITÉ DE SHERBROOKE
Faculté de génie
Département de génie électrique et de génie informatique

ARCHITECTURE GÉNÉRIQUE
MULTICOUCHE POUR L'ANALYSE DE
PERFORMANCES MUSICALES
POLYPHONIQUES

Thèse de doctorat
Spécialité : génie électrique

Bruno GAGNON

Sherbrooke (Québec) Canada

Décembre 2019

MEMBRES DU JURY

Roch LEFEBVRE

Directeur

Charles-Antoine BRUNET

Codirecteur

Martin BOUCHARD

Évaluateur

Philippe-Aubert GAUTHIER

Évaluateur

Philippe MABILLEAU

Évaluateur

RÉSUMÉ

Cette thèse de doctorat, du domaine de la recherche applicative, propose un système informatique destiné au développement d'applications logicielles s'adressant aux musiciens. Le système proposé permet essentiellement de suivre un musicien en meilleur effort à travers une partition de musique, et ce, tout en évaluant les erreurs de performance qu'il a réalisées telles que des notes jouées trop tôt ou trop tard et des notes manquantes. Pour ce faire, le système utilise en entrée le signal audio numérique du jeu du musicien et la partition de musique en format numérique, soit le format MusicXML.

Comparativement aux systèmes existants, le système proposé ne nécessite aucun entraînement et donne la liberté au musicien de jouer n'importe quel segment de la partition de musique sans avoir à préalablement informer le système de ses intentions. Ainsi, le système proposé a la capacité de retrouver le musicien partout à travers la partition de musique, et ce, pourvu que le musicien joue suffisamment de notes. Cette liberté permet au musicien de se concentrer sur l'exécution de la performance musicale puisque c'est l'application qui s'adapte au musicien et non le contraire.

Le cœur du système proposé est composé de quatre couches, soit la couche de traitement du signal, la couche de transcription automatique, la couche d'alignement et la couche comparative. Le présent document révisé d'abord l'état de l'art concernant le traitement du signal numérique et concernant les méthodes d'alignement, et ce, en lien avec le domaine de la musique. Par la suite, ce document présente en détail le fonctionnement ainsi que la performance de chacune des couches au cœur du système proposé.

Mots-clés : Traitement du signal, Transcription automatique, Suivi automatique de partition, Musique

REMERCIEMENTS

Je tiens à exprimer toute ma reconnaissance à mon directeur Monsieur Roch Lefebvre, Professeur à l'Université de Sherbrooke, qui a été une pierre angulaire à la réalisation de ce projet de doctorat de par son soutien moral et financier, son encadrement et sa bienveillance. Je tiens à mentionner que M. Lefebvre a eu un impact significatif sur le développement de ma rigueur scientifique, de mon esprit critique et de mon autonomie, soit des atouts qui, aujourd'hui, me permettent de profiter pleinement de ma carrière professionnelle. Ceci dit, je me considère privilégié d'avoir côtoyé M. Lefebvre comme, au-delà de son rôle d'enseignant, il a été pour moi un modèle à suivre de par ses talents en communication, de son leadership et de sa grande générosité.

J'aimerais aussi exprimer ma sincère gratitude envers mon codirecteur Monsieur Charles-Antoine Brunet, Professeur à l'Université de Sherbrooke, qui a su apporter à ce projet de recherche les fondements nécessaires à sa réussite de par sa pensée structurée et de par son excellente maîtrise du design de systèmes informatiques. Par la qualité de ses explications, M. Brunet a su fortifier mes connaissances et ma confiance en programmation pour me permettre de réaliser l'ensemble du code en appui à ce projet de recherche. Je tiens aussi à remercier M. Brunet pour sa disponibilité, pour la qualité des divers entretiens et, plus particulièrement, pour ses divers encouragements, sa cordialité et sa bonté pour lesquels je garderai toujours d'excellents souvenirs.

Je voudrais également remercier l'ensemble du personnel du Groupe de recherche sur la parole et l'audio (GRPA) pour leur aide et support. Je tiens aussi à témoigner toute ma gratitude au Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG) pour son soutien financier. Un grand merci également à mon employeur le Centre de services partagés du Québec (CSPQ) pour son aide, son soutien et ses encouragements qui m'ont permis de finaliser mes études de doctorat.

J'aimerais aussi remercier les membres du jury pour leurs excellents commentaires, corrections et suggestions d'amélioration, soit plus précisément, le professeur Monsieur Philippe Mabillean de l'Université de Sherbrooke, le professeur Monsieur Martin Bouchard de l'Université d'Ottawa et le professeur Monsieur Philippe-Aubert Gauthier de l'Université de Québec à Montréal (UQÀM). Il est aussi important pour moi de remercier spécialement Monsieur Christian Lachapelle, coordonnateur académique aux études supérieures, pour son excellent travail de coordination.

Pour terminer, j'aimerais témoigner ma gratitude à ma très chère conjointe pour ses judicieux conseils, sa patience et son support qu'elle a su me donner tout au long de ce projet. Merci aussi à Madame Jacqueline Garriss et ses étudiants pour les enregistrements audio de violon utilisés pour certaines des évaluations. Finalement, j'adresse mes sincères remerciements à l'ensemble du personnel de la faculté de génie pour leurs multiples aides et assistances ainsi que pour leur professionnalisme et courtoisie.

TABLE DES MATIÈRES

1	INTRODUCTION	1
1.1	Mise en contexte et problématique	1
1.2	Définition du projet de recherche	2
2	ÉTAT DE L'ART - TRAITEMENT DU SIGNAL	5
2.1	Algorithmes d'estimation de tonies multiples (AETM)	5
2.1.1	Algorithmes basés sur les filtres en peignes	6
2.1.2	Algorithmes basés sur la transformée de Fourier rapide	8
2.1.3	Algorithmes basés sur la psychoacoustique	11
2.2	Détections des transitions	12
2.2.1	Algorithmes basés sur les paramètres temporels	12
2.2.2	Algorithmes basés sur les paramètres fréquentiels	12
2.2.3	Paramètres temporels et fréquentiels jumelés	13
3	ÉTAT DE L'ART - MÉTHODES D'ALIGNEMENT	15
3.1	Méthodes basées sur les HMM	15
3.2	Modèles bayésiens	20
3.3	Déformation dynamique du temps	21
3.4	EDTW	26
4	APPLICATIONS EXISTANTES	35
4.1	SmartMusic	35
4.2	Professeur virtuel de musique	38
4.3	VEMUS	41
4.4	Digital Violin Tutor (DVT)	42
4.5	Tonara	45
5	SYSTÈME PROPOSÉ	49
5.1	Module <i>Saisie du signal audio</i>	49
5.2	Module <i>Saisie de la partition de musique</i>	50
5.3	Module <i>Analyse de la performance musicale</i>	50
5.3.1	Couche de traitement du signal	51
5.3.2	Couche de transcription automatique	51
5.3.3	Couche d'alignement	53
5.3.4	Couche comparative	55
5.3.5	Architecture de type <i>bottom-up</i>	55
5.4	Module <i>Application</i>	56
6	COUCHE DE TRAITEMENT DU SIGNAL	57
6.1	Instrument de musique visé par la couche	59
6.1.1	Inharmonicité des signaux de violon	59

6.1.2	Polyphonie des signaux de violon	61
6.2	Fichiers audio utilisés et critères d'évaluation	62
6.3	Algorithme de base utilisé pour réaliser la couche	63
6.4	L'estimateur de tonies proposé	64
6.4.1	Calculer la somme harmonique	64
6.4.2	Rechercher les tonies potentielles	69
6.4.3	Calculer le niveau de confiance des tonies	72
6.4.4	Sélectionner les meilleurs candidats	72
6.5	Pondération des tonies potentielles	72
6.5.1	Confiance sur l'intensité	72
6.5.2	Confiance sur l'octave	73
6.5.3	Confiance sur le pic	74
6.5.4	Paramètres testés	74
6.5.5	Les meilleurs paramètres	76
6.6	Discussion	78
6.7	Contributions scientifiques de la couche	79
7	COUCHE DE TRANSCRIPTION AUTOMATIQUE	81
7.1	Mise en contexte	83
7.2	L'estimateur de notes proposé	85
7.2.1	Étape : <i>concatener sous condition</i>	87
7.2.2	Étape : <i>Fusionner les adjacents</i>	89
7.2.3	Retirer les mauvais	92
7.2.4	Étape : <i>Créer des nouveaux</i>	93
7.2.5	Retirer les complets	93
7.3	Rejet d'une note selon son intensité sonore	93
7.4	Attribution d'un niveau de confiance aux notes estimées	95
7.4.1	Niveau de confiance pour le début et la fin d'une note	95
7.4.2	Niveau de confiance des notes selon l'octave	101
7.4.3	Niveau de confiance d'une note selon les tonies la constituant	102
7.5	Discussion	102
7.6	Contributions scientifiques de la couche	103
8	COUCHE D'ALIGNEMENT	105
8.1	Aperçu du système d'alignement proposé	106
8.2	Alignement brut	106
8.2.1	Éléments de l'algorithme	107
8.2.2	Étapes de fonctionnement	119
8.2.3	Délai d'expiration	136
8.2.4	Justification de l'algorithme EDTW	137
8.3	Alignement fin	139
8.3.1	Notes jouées en polyphonie	144
8.3.2	Notes potentielles de même départ et de même identité	144
8.3.3	Correction d'imperfections	144
8.4	Contributions scientifiques de la couche	146

9	COUCHE COMPARATIVE	149
9.1	Difficultés reliées aux imperfections des couches inférieures	149
9.2	Définition des variables	150
9.3	Estimation du tempo	150
9.4	Déviaton sur le moment de l'attaque	151
9.5	Déviaton sur la durée	152
9.6	Justesse de l'intonation	153
9.7	Notes manquantes	155
10	PERFORMANCE : COUCHE DE TRAITEMENT DU SIGNAL	157
10.1	Performance pour des signaux monophoniques	157
10.2	Performance pour des signaux polyphoniques	159
10.3	Temps de traitement	160
10.4	Discussion	161
11	PERFORMANCE : COUCHE DE TRANSCRIPTION AUTOMATIQUE	163
11.1	Apport de la couche par rapport à l'approche typique	163
11.2	Décomposition en plusieurs notes	164
11.3	Rejet d'une note selon son intensité sonore	165
11.4	Performance des niveaux de confiance de note	166
11.5	Performance des trois niveaux de confiance combinés	167
11.6	Temps de traitement	169
11.7	Discussion	170
12	PERFORMANCE : COUCHE D'ALIGNEMENT	171
12.1	Performance - partitions monophoniques	171
12.2	Performance - partitions polyphoniques	174
12.3	Performance pour des segments joués de façon isolés	175
12.4	Performance de la fenêtre adaptative de calcul	177
12.5	Discussion	178
13	PERFORMANCE DE LA COUCHE COMPARATIVE	181
13.1	Discussion	183
14	CONCLUSION	185
14.1	Sommaire des travaux réalisés	185
14.2	Sommaire des résultats obtenus	186
14.3	Contributions	187
14.4	Retour sur les objectifs fixés et la question de recherche	188
14.5	Travaux futurs	188
A	MODÈLES DE MARKOV CACHÉS	191
B	DÉFORMATION DYNAMIQUE DU TEMPS	195
C	VALEURS DE CONFIANCE D'UN CS	197

C.1	Collecte des données pertinentes	199
C.2	Calcul du tempo	200
C.3	Calcul du niveau de confiance relatif au moment de début	200
C.4	Calcul du niveau de confiance relatif aux durées	201
C.5	Calcul du niveau de confiance relatif à la consistance du tempo	201
C.6	Calcul du niveau de confiance relatif à la proximité du tempo	202
C.7	Calcul du niveau de confiance relatif à l'intensité des notes	202
C.8	Calcul du niveau de confiance global de l'élément	203
D	PROCÉDURE POUR OBTENIR LA TONIE NOMINALE	205
E	DÉTAILS DE PERFORMANCE DE LA COUCHE COMPARATIVE	207
	LISTE DES RÉFÉRENCES	211

LISTE DES FIGURES

2.1	Exemple de filtre en peigne	7
2.2	Configuration en cascade de filtres en peigne	7
3.1	Exemple d'un HMM utilisé modélisant une note	16
3.2	HMM plus performant pour une note de courte durée	17
3.3	HMM permettant de modéliser les notes jouées de différentes façons	17
3.4	Modélisation de la partition note par note	18
3.5	Modélisation de la partition avec des notes et des états fantômes	18
3.6	Exemple de paramètres observables pour des notes de piano	18
3.7	Modèle bayésien permettant de modéliser le tempo	20
3.8	Exemple d'alignement obtenu par le DTW.	22
3.9	Exemple d'un spectre attendu et réel pour la méthode PSD	24
3.10	Exemple de matrice de similarités	27
3.11	Exemple de chemins survivants et d'alignement	28
3.12	Exemple d'un bloc $M \times N$ à l'intérieur de la matrice de similarités	29
3.13	Déplacements possibles pour un chemin local (CL)	29
3.14	Tous les CL possibles pour un bloc de dimension 3×3	30
3.15	Exemple de matrice de chemins survivants	32
3.16	CS après le retrait des CS ne respectant pas les spécifications minimales	33
3.17	CS sélectionné pour l'alignement	34
4.1	Fenêtre principale du logiciel SmartMusic de MakeMusic	36
4.2	Exemple d'erreurs de performance SmartMusic lors d'une désynchronisation	37
4.3	Exemple d'erreurs de performance SmartMusic pour un segment très rapide	38
4.4	Fenêtre principale de l'application PVM	39
4.5	Exemple d'erreurs de performance trouvées par l'application PVM	40
4.6	Fenêtre principale du logiciel IMUTUS [67]	42
4.7	Fenêtre principale de l'application DVT	43
4.8	Exemple d'alignement obtenu par DVT	44
4.9	Exemple de résultats obtenus par l'AETM utilisé par l'application DVT	44
4.10	Capture d'écran de l'application Tonara	46
5.1	Modules de système	49
5.2	Schéma bloc global du système	50
5.3	Exemple de trames chevauchées	51
5.4	Exemple de tonies estimées	52
5.5	Notes de musiques jouées	52
5.6	Exemple de notes estimées	53
5.7	Exemple d'alignement estimé par la couche d'alignement	54
6.1	Exemple de trames chevauchées	57
6.2	Exemple de tonies estimées par la couche traitement du signal	58

6.3	Spectre en fréquence de la corde ouverte MI (659.5 Hz) jouée en pizzicato	60
6.4	Spectre en fréquence de la corde ouverte MI (659.5 Hz) jouée avec l'archet	60
6.5	Problème d'estimation de tonies avec l'algorithme Yin	62
6.6	Portion du signal utilisé pour l'évaluation de la performance	63
6.7	Harmsum d'un signal harmonique où f0 correspond à la raie d'indice 5	65
6.8	Harmsum d'un signal harmonique où f0 correspond à la raie d'indice 5.2	67
6.9	Harmsum d'un signal harmonique où f0 correspond à la raie d'indice 5.3	67
6.10	Performance pour différentes tonies	69
6.11	Performance de l'estimateur pour violon vs. référence	70
6.12	Organigramme utilisé pour estimer les tonies	71
6.13	Courbes de pondération psychoacoustique	75
6.14	Exemple d'histogramme du niveau de confiance	76
6.15	Distribution normalisée du niveau de confiance	78
6.16	Performance pour les niveaux de confiance	79
7.1	Exemple de tonies estimées	81
7.2	Exemple de notes estimées	82
7.3	Exemple de tonies estimées	83
7.4	Exemple de trames chevauchées	83
7.5	Exemple de tonies estimées à l'extérieur de la plage attendue	84
7.6	Notes résultantes en utilisant la méthode typique	85
7.7	Exemple de tonies manquantes	85
7.8	Étapes de la procédure	86
7.9	Segment avec les conditions 1-2 de la <i>Concaténation conditionnelle</i>	88
7.10	Exemple de segments multiples par note en présence de tonies manquantes	90
7.11	Exemple de segments multiples en présence d'un vibrato	90
7.12	Exemple de tonies interpolées avec l'interpolation cubique à quatre points	92
7.13	Courbes de gain isosonique	94
7.14	Exemple de tonies estimées après le relâchement de la note	96
7.15	Intensité de la 2 ^e note de la figure 7.14 avec mauvaise fin	97
7.16	Procédure pour trouver toutes les fins possibles de note	98
7.17	Exemple de décomposition en plusieurs notes	99
7.18	Notes découlants de la note illustrée à la figure 7.17	100
8.1	Exemple d'alignement	105
8.2	Schéma bloc de la couche d'alignement	106
8.3	Exemple des notes jouées en polyphonie	107
8.4	Exemple fictif de matrice de similarité utilisée par l'algorithme EDTW	108
8.5	Contraintes de déplacement de l'algorithme EDTW	108
8.6	Exemple de notes potentielles pour créer l'axe des notes détectées	109
8.7	Axe d'index (j) d'un ensemble de notes détectées à départ commun	110
8.8	Exemple de matrice de similarité utilisé par l'alignement brut	112
8.9	Exemple de chemins survivants pour l'algorithme EDTW	112
8.10	Contraintes de déplacement de l'alignement brut	113
8.11	Exemple fictif de chemins survivants	114

8.12	Exemple d'alignement obtenu à l'aide de la fenêtre adaptative de calcul . . .	118
8.13	Différentes étapes de l'alignement brut	119
8.14	Exemple de matrice de similarité avec des éléments oubliés	120
8.15	Exemple de gabarit pour le centre de la fenêtre adaptative de calcul	121
8.16	Illustration des variables utilisées par le pseudo-code de l'algorithme 1 . . .	123
8.17	Exemple de chemins potentiels obtenus à partir de la figure 8.16	124
8.18	Exemple fictif de CS à l'intérieure d'une matrice de similarité	125
8.19	Matrice de la figure 8.18 à la suite d'une mise à jour	126
8.20	CS crédible à partir de la matrice de similarité de la figure 8.19	126
8.21	Exemple fictif de trois chemins survivants	128
8.22	Exemple de positionnement de blocs d'analyse de dimensions 4×4	130
8.23	Combinaisons possibles de fusionnement à partir des CS de la figure 8.22 .	130
8.24	Exemple de chemins survivants lorsque joué du début jusqu'à la fin	131
8.25	Exemple fictif de chemins survivants lorsque joué en désordre	132
8.26	Positionnement des limites de fusionnement	132
8.27	Exemple de calcul de la valeur de confiance d'un alignement brut	134
8.28	Représentation d'une aire protégée d'une matrice de similarité	136
8.29	Exemple de segment de partition répétitif	136
8.30	Extrait d'une matrice de similarité en présence d'un segment répétitif . . .	137
8.31	Représentation en arbre des combinaisons possibles de notes de la partition	140
8.32	Représentation en arbre de toutes les possibilités	141
8.33	Plages temporelles où le début des notes de la partition est attendu	142
8.34	Représentation en arbre simplifié selon les contraintes temporelles	143
8.35	Représentation en arbre minimisant le nombre de notes manquantes	143
8.36	Représentation en arbre d'un exemple avec polyphonie	145
8.37	Plage temporelle pour la correction d'imperfections	146
9.1	Variables utilisées par la couche comparative	151
9.2	Courbe de pondération utilisée pour estimer la tonie d'une note courte . .	154
10.1	En monophonie, tonies estimées selon le seuil sur le niveau de confiance . .	158
10.2	En polyphonie, tonies estimées selon le seuil sur le niveau de confiance . . .	158
11.1	Condition pour une note estimée correctement	164
11.2	Condition plus restreinte pour une note estimée correctement	165
11.3	Performance du niveau de confiance de début et de fin	167
11.4	Performance du niveau de confiance sur l'octave	168
11.5	Performance du niveau de confiance selon les tonies	169
12.1	Exemple de segment répétitif contenu dans la partie <i>Var II</i>	173
A.1	Exemple d'un HMM utilisé pour reconnaître une note de musique	192
A.2	Densités de probabilités pour le HMM de la figure A.1	192
B.1	Deux exemples d'avancement à travers la matrice de similarités	196
C.1	Exemple fictif de matrice de similarité	198

C.2 Exemple de chemin survivant composé de 7 éléments 199

LISTE DES TABLEAUX

6.1	Pourcentage de déviation de l'harmonique	61
6.2	Caractéristiques de l'enregistrement audio	62
6.3	Performance et meilleurs paramètres pour diverses transformées	68
6.4	Paramètres testés	68
6.5	Meilleurs paramètres pour le calcul du niveau de confiance	77
8.1	Exemple d'axe (i) de notes à départ commun au niveau de la partition . . .	108
8.2	Début des notes pour le premier chemin survivant	115
8.3	Début des notes pour le deuxième chemin survivant	115
8.4	Exemple de notes représentant une partition de musique	139
8.5	Exemple de notes potentielles détectées	140
8.6	Plages temporelles où le début des notes de la partition est attendu	142
8.7	Exemple de notes d'une partition avec des notes jouées en polyphonie . . .	144
9.1	Définition des variables utilisées par la couche comparative	150
9.2	Durée d'une note jouée en fonction de l'accent	153
10.1	Performance de l'estimateur de tonies pour des signaux monophoniques . . .	159
10.2	Performance de l'estimateur de tonies pour des signaux polyphoniques . . .	160
11.1	Performance de la typique versus proposée avec et sans fusionner adjacents	164
11.2	Impact de la décomposition en plusieurs notes	165
11.3	Impact du rejet selon l'intensité	166
11.4	Combinaison des trois niveaux de confiance	168
12.1	Performance de la couche d'alignement pour des partitions monophoniques	172
12.2	Performance de la couche d'alignement pour des partitions polyphoniques .	174
12.3	Performance de l'alignement lorsque 30 notes sont jouées de façon isolée . .	176
12.4	Résultats avec la fenêtre adaptative de calcul	177
13.1	Compilation des résultats de performance de la couche comparative	182
A.1	Variables d'un HMM	191
A.2	Séquence d'observations du problème pour l'exemple sur le DTW	193
B.1	Données pour l'exemple sur le DTW	195
B.2	Exemple d'un tableau utilisé pour les DTW pour démontrer les calculs . . .	195
B.3	Matrice de similarité	196
C.1	Exemple d'axe (j) de notes détectées à départ commun	197
C.2	Exemple d'axe (i) de notes à départ commun au niveau de la partition . . .	198
C.3	Information pertinente du CS de la figure C.2	199
D.1	Exemples de gammes naturelles et ratio pour obtenir les f0 nominales	206

E.1	Performance de la couche comparative avec l’Hongrie	208
E.2	Performance de la couche comparative avec l’Andantino	209

CHAPITRE 1

INTRODUCTION

1.1 Mise en contexte et problématique

« *Practice makes perfect* »

Certes, un grand musicien naît de la pratique, mais la pratique individuelle d'un instrument de musique est une activité dont la qualité du travail exécuté dépend de plusieurs facteurs tels que le niveau de motivation, l'expérience et la concentration. Malheureusement, seul avec un instrument, la motivation n'est pas garantie. De plus, sans un guide, l'expérience n'est pas nécessairement au rendez-vous. Reste la concentration, mais sans l'expérience, la concentration est-elle dirigée sur l'important ? Sinon, est-ce que le musicien perfectionne des défauts en filtrant l'important ? Lors de ses pratiques individuelles, l'idéal pour un musicien apprenti serait probablement de se faire accompagner d'un professeur de musique afin de se faire guider et motiver, mais, de toute évidence, cette dernière option est inabordable pour la majorité d'entre nous.

Il y a tout de même de l'espoir, car certaines solutions réalistes pourraient probablement être développées dans un futur rapproché. Par exemple, imaginons que les exercices techniques monotones de la pratique individuelle soient remplacés par des jeux vidéos motivants tels que *Guitar Hero* [62] ou imaginons que l'expérience soit simulée par un logiciel réalisant un professeur virtuel de musique. Afin de rendre ces exemples d'applications possibles pour des musiciens débutants et avancés, il est nécessaire de se poser la question de recherche suivante :

Est-ce qu'il est possible pour un système informatique de suivre de façon autonome, en temps réel et sans entraînement une prestation musicale de niveau avancée, c'est-à-dire composée de notes rapides et polyphoniques et, ensuite, d'identifier les erreurs par rapport à la partition de musique jouée?

Actuellement, il existe essentiellement deux applications permettant le suivi d'une prestation musicale et l'identification d'erreurs de performance musicale, c'est-à-dire SmartMusic [5] et Tonara [50] respectivement. Cependant, ces applications sont limitées et donnent peu de liberté au musicien. Par exemple, l'application Tonara permet seulement de tourner les pages d'une partition de musique, et ce, pourvu que le musicien joue du début

jusqu'à la fin sans trop d'erreurs. Pour sa part, SmartMusic impose le suivi rigoureux d'un métronome sans quoi il sera désynchronisé et donnera un nombre important d'erreurs de performance sans rapport avec la réalité.

Afin de créer des applications sophistiquées donnant plus de liberté aux musiciens, il est nécessaire de faire quelques avancements technologiques. C'est d'ailleurs l'objectif de ce projet de recherche. Plus précisément, l'objectif est de développer une architecture logicielle et des algorithmes pour permettre la réalisation d'applications logicielles permettant de suivre un musicien jouant librement une partition de musique connue du système, et ce, tout en détectant automatiquement les erreurs de performance musicale du musicien. Pour ce faire, la performance du musicien sera captée par un microphone et traitée par un ordinateur personnel afin de déceler les erreurs telles que celles sur la durée et sur la tonalité d'une note. Il est à noter que le tout sera réalisé sans aucun entraînement préalable du système.

1.2 Définition du projet de recherche

Ce projet de doctorat permettra l'identification des modules nécessaires pour la création d'applications musicales sophistiquées telles qu'un professeur virtuel de musique. Ce projet permettra aussi la création d'algorithmes innovants de traitement du signal, et ce, plus spécifiquement pour des signaux audio de musique à caractère polyphonique¹.

L'objectif principal de ce projet de recherche est de concevoir une architecture logicielle et des algorithmes permettant le développement d'applications musicales capables de suivre efficacement un musicien à travers une partition de musique tout en lui donnant la liberté de jouer la partition dans l'ordre qu'il le désire. Le tout devra fonctionner sans entraînement, c'est-à-dire que le système n'aura pas besoin d'être entraîné avec de longues bases de données pour être en état de marche. Le tout devra aussi fonctionner avec des partitions de musique complexes, soit des partitions contenant des segments de notes rapides et des segments de notes jouées en polyphonie.

Comme objectifs complémentaires, ce projet de doctorat proposera des stratégies permettant la détection automatique d'erreurs de performance musicale telles que des notes jouées trop tôt ou trop tard. Il permettra aussi la conception d'un estimateur de notes de musique où les notes seront estimées à partir des tonies² estimées par un algorithme d'estimation polyphonique de tonies.

1. La polyphonie est une organisation de notes qui se superposent dans le temps, et ce, comparativement à la monophonie, où les notes sont jouées une à la suite de l'autre.

2. La tonie est une caractéristique des notes qui permet la classification sur une échelle en fréquence.

La principale contribution scientifique de ce projet de recherche est la création d'un suiveur de partition offrant davantage de liberté aux musiciens, c'est-à-dire que, comparativement aux suiveurs existants, ce dernier pourra suivre un musicien jouant une partition dans un ordre et avec le tempo quelconque, et ce, sans que le musicien n'ait à informer préalablement le système de ses intentions. Bien attendu, le musicien devra jouer suffisamment de notes pour que le système puisse associer ces dernières aux bonnes notes de la partition.

Comme contributions scientifiques secondaires, ce projet de recherche a permis la création d'algorithmes ayant pour mission de peupler une liste de notes potentiellement jouées par un musicien, et ce, à partir du signal audio correspondant au jeu de ce dernier. Il est à noter que les algorithmes développés estiment les notes jouées malgré la présence de vibrato, de tonies manquantes et de notes dont la justesse varie en dehors des limites attendues.

Enfin, comme autres contributions scientifiques secondaires, ce projet a aussi permis la création d'un estimateur de tonies optimisé spécifiquement pour des pièces complexes de violon. Il est à noter que le violon a été choisi pour cet estimateur comme les notes peuvent être jouées très rapidement, avec ou sans un vibrato, et ce, avec des styles variés tels que le staccato, le pizzicato et le détaché. De plus, le violon est considéré comme légèrement polyphonique puisqu'il permet de jouer un maximum de quatre notes en simultanée.

Vous retrouverez dans les chapitres qui suivent, l'état de l'art concernant les algorithmes de traitement du signal dédiés à la reconnaissance de la musique, l'état de l'art concernant les méthodes d'alignement, c'est-à-dire les techniques permettant de situer chacune des notes de la partition à chacun des emplacements temporels du signal audio du jeu du musicien correspondant, et ce, afin de pouvoir le situer dans la partition de musique. Ensuite, un aperçu des applications de musique en lien avec ce projet sera présenté.

Une fois l'état de l'art révisé, le fonctionnement du système proposé sera présenté globalement et, ensuite, en détail. Le détail sera présenté en plusieurs chapitres, dont chacun correspondra à une couche logicielle du système. La description de chacune des autres couches sera suivie par quatre chapitres correspondants aux résultats de performance obtenus pour chacune de quatre couches. Pour sa part, le dernier chapitre de la thèse permettra d'établir une conclusion sur les travaux de recherches réalisés.

CHAPITRE 2

ÉTAT DE L'ART - TRAITEMENT DU SIGNAL

Dans le but de répondre à la question de recherche, il est nécessaire de maîtriser certains algorithmes de traitement du signal numérique. Ce chapitre présente les deux familles d'algorithmes jugées pertinentes pour ce type de système informatique, c'est-à-dire des algorithmes permettant d'estimer les multiples tonies contenues dans un signal audio numérique et des algorithmes permettant d'estimer les transitions contenues dans un signal audio numérique.

Il est à noter que ces deux familles d'algorithmes permettent de récupérer l'information nécessaire à l'estimation des notes contenues dans un signal audio numérique. En effet, les algorithmes d'estimation de tonies multiples permettent de retrouver la fréquence de la composante fondamentale de chacune des notes et les algorithmes d'estimation de transitions permettent de retrouver la position temporelle de début de chacune des notes. Les deux prochaines sections présentent plusieurs algorithmes usuels, et ce, pour chacune de ces familles d'algorithme.

2.1 Algorithmes d'estimation de tonies multiples (AETM)

La fréquence fondamentale f_0 d'une tonalité harmonique complexe¹ est généralement une bonne approximation de la tonie (*pitch*) [56] perçue. Plus précisément, la distance entre les harmoniques adjacentes d'un signal audio est généralement la tonie perçue et correspond à la fréquence fondamentale [49].

Il est à noter que la fréquence fondamentale n'a pas besoin d'être présente pour que la tonie perçue d'une tonalité harmonique complexe corresponde à cette dernière, la plus célèbre des théories tentant d'expliquer ce phénomène a été proposée par Ernst Terhardt [79] et est mieux connue comme étant le concept de tonie virtuelle (*concept of virtual pitch*, en anglais). Selon cette théorie, le cerveau chercherait, par apprentissages, à reconnaître les relations harmoniques des composantes contenues dans les tonalités harmonique complexe, et ce, pour leur associer une fréquence fondamentale.

Les premiers algorithmes d'estimation de la tonie ont été développés dans les années 1970. La référence [53] en est un exemple où la fréquence fondamentale est trouvée en addition-

1. Dans le cadre de cette thèse, une *tonalité harmonique complexe* correspond au son d'une note ayant un timbre particulier et générée par un instrument de musique.

nant au spectre fréquentiel originel plusieurs spectres fréquents où l'axe des fréquences est compressé par des facteurs de 2,3,4, etc., et en recherchant ensuite la fréquence d'amplitude maximum correspondant à la fréquence fondamentale estimée.

Aujourd'hui, il existe des centaines d'articles scientifiques proposant des algorithmes d'estimation de la tonie. Jusqu'ici, il n'y a pas de solution suffisamment performante pour résoudre tous les cas, et ce, notamment pour des signaux de musique contenant plusieurs notes jouées en simultanées, c'est-à-dire des signaux polyphoniques. C'est pourquoi il y a encore beaucoup de chercheurs qui se penchent sur le sujet. Cette abondance de publications a l'avantage de fournir plusieurs idées d'approches sur lesquelles les chercheurs peuvent se baser pour résoudre leur cas particulier.

L'estimation de plusieurs tonies dans un signal polyphonique est une tâche complexe puisqu'un pic contenu dans le module du spectre peut être formé de plusieurs harmoniques superposées. La complexité de cette tâche contraint les chercheurs à produire des algorithmes d'estimation de tonies multiples (AETM) très complexes intégrant souvent plusieurs méthodes de traitement du signal [35]. Par conséquent, il est très difficile de catégoriser les AETM en différentes classes. Comme l'estimation des tonies contenues dans un signal de musique est une tâche importante du système qui sera développé durant ce sujet de recherche, les sous-sections qui suivent présentent quelques approches utilisées pour produire des AETM telles que les approches utilisant des filtres en peignes, la transformée de Fourier rapide et des propriétés de la psychoacoustique.

2.1.1 Algorithmes basés sur les filtres en peignes

Certains AETM utilisent des filtres en peigne (*comb filters*) définis selon l'équation 2.1 où la variable g permet de modifier le gain du filtre [23]. La valeur de m est choisie afin de filtrer une fréquence fondamentale f_0 et ses harmoniques f_k selon l'équation 2.2 où f_s et $[\cdot]$ représentent respectivement la fréquence d'échantillonnage et un arrondissement à l'unité [83]. La figure 2.1 illustre un exemple de filtre en peigne où les variables g et m valent 1 et 9 respectivement.

$$Y(z) = X(z)(1 - gz^{-m}) \quad (2.1)$$

$$m = \lceil f_s / f_0 \rceil \quad (2.2)$$

Ces AETM sont généralement composés de plusieurs filtres en peigne placés en cascades [47] ou en parallèle [77]. La figure 2.2 illustre un exemple de configuration en cas-

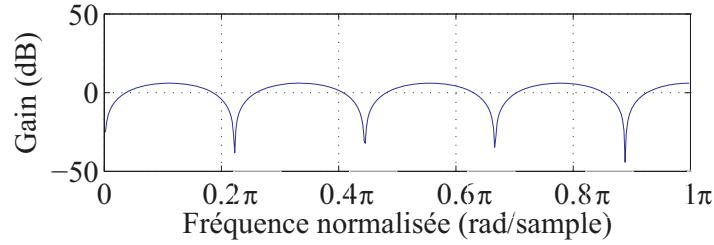


Figure 2.1 Exemple de filtre en peigne

cade [47] où chacun des 12 filtres permet de retirer la fréquence fondamentale et les harmoniques de l'une des 12 notes d'une octave n . Dans cet exemple, les tonies sont trouvées de façon itérative en recherchant d'abord la sortie du filtre $y_x[n]$ qui donne un signal de sortie d'énergie nulle, en modifiant ensuite la configuration pour mettre ce filtre en premier dans la cascade de 12 filtres et en effectuant le filtrage de nouveau avec la nouvelle cascade de filtres. À la suite de plusieurs itérations, les premiers filtres de la cascade donnant ensemble une énergie nulle correspondent aux notes contenues dans le signal polyphonique.

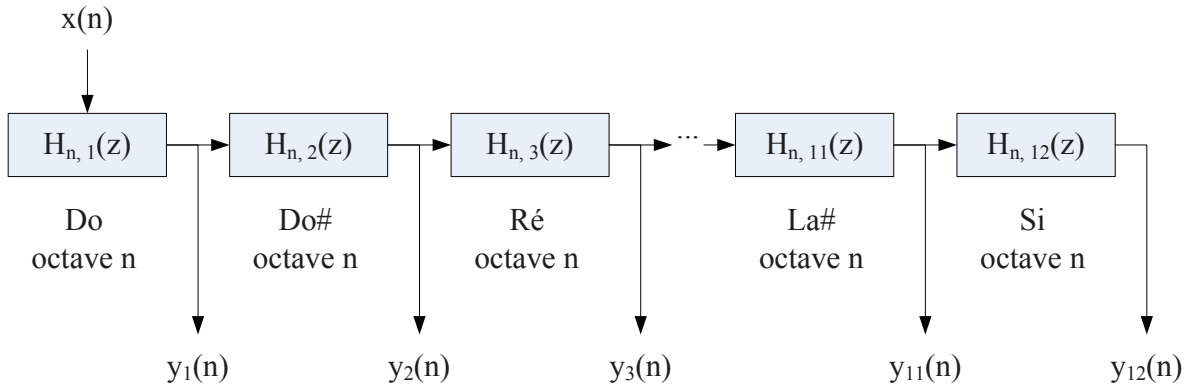


Figure 2.2 Configuration en cascade de filtres en peigne [47]

Quelle que soit la configuration utilisée, lorsque le nombre de filtres en cascade augmente, le signal de sortie des filtres se déforme progressivement jusqu'à un point où les algorithmes ne fonctionnent plus. Ce phénomène s'explique par le fait que la réponse en fréquence des filtres en peignes est très arrondie et éloignée d'une réponse en fréquence idéale. Pour améliorer les résultats, l'article [23] propose d'ajouter un pôle près de chacun des zéros de l'équation 2.1 pour aplatir la bande passante selon l'équation 2.3 où plus la variable p s'approche de la valeur 1, plus la bande passante s'élargit et s'aplatit.

$$Y(z) = \frac{1 - z^{-m}}{1 - p^m z^{-m}} \quad (2.3)$$

Pour tous les AETM basés sur les filtres en peigne, les fréquences les plus atténuées des filtres sont centrées sur la fréquence fondamentale nominale d'une note bien précise et aux multiples de cette fréquence. Avec des signaux réels, les notes peuvent être mal jouées ou jouées avec des effets tels que des vibratos et, par conséquent, les fréquences fondamentales ne correspondent souvent pas exactement aux fréquences attendues. En conclusion, la performance des AETM basés sur les filtres en peigne n'est pas optimale pour des instruments de musique réels.

2.1.2 Algorithmes basés sur la transformée de Fourier rapide

Les AETM basés la transformée de Fourier rapide (*Fast Fourier transform*, FFT) sont les plus fréquents [55]. Ces algorithmes décomposent d'abord le signal en trames afin de faire des analyses locales. Généralement, comme la fréquence fondamentale n'est pas toujours très apparente dans le spectre, ces AETM utilisent aussi l'amplitude des harmoniques pour faire les estimations. Ces AETM ont parfois de la difficulté à estimer les notes graves, car la résolution du spectre fréquentiel est parfois trop faible pour évaluer correctement des fréquences très graves, et ce, même avec une interpolation.

Lorsque la FFT est effectuée sur un segment audio stationnaire contenant plusieurs notes simultanées, les pics contenus dans le module du spectre peuvent être le résultat de la superposition d'une ou de plusieurs harmoniques. Cette superposition d'harmoniques complexifie l'estimation des notes jouées. Ainsi, certains AETM estiment d'abord la note la plus probable et soustraient ensuite au module du spectre une estimation du module qui appartiendrait à cette même note si elle était jouée seule.

Ensuite, ces AETM estiment une nouvelle note à partir du module allégé et retirent de nouveau le module estimé de cette note détectée, et ce, jusqu'à ce que toutes les notes soient détectées [17, 38].

Habituellement, ces algorithmes sont performants pour des instruments de musique où le musicien n'a le contrôle que sur la dynamique de l'attaque des notes comme le piano puisque l'enveloppe spectrale est relativement constante pour une note donnée. Par contre, pour certains instruments comme le violon, où le musicien peut contrôler plusieurs autres paramètres, l'enveloppe spectrale est plus difficile à estimer. Par conséquent, ces algorithmes ne sont pas performants pour tous les types d'instruments.

Les notes peuvent être jouées avec des effets tels qu'un vibrato ou peuvent être jouées avec peu de justesse. Par conséquent, la fréquence fondamentale d'une note d'indice n n'est pas nécessairement égale à la fréquence fondamentale nominale attendue de la note telle que

définie par l'équation 2.4 pour le format MIDI (Musical Instrument Digital Interface) où $n = 69$ pour la note La de la 4^e octave d'un piano (La4)².

$$f_n = 440 \times 2^{\frac{n-69}{12}} \quad (2.4)$$

Néanmoins, si l'instrument est correctement accordé, les fréquences fondamentales sont généralement autour de la fréquence fondamentale nominale, c'est-à-dire dans une plage de plus ou moins 50 cents³ puisque les notes sont habituellement séparées entre elles par une distance constante d'environ 100 cents⁴. L'équation 2.5 permet de convertir une distance entre une fréquence f_a et une fréquence f_b en cents.

$$d_{cents} = 1200 \log_2(f_b/f_a) \quad (2.5)$$

D'ailleurs, l'algorithme utilisé par le projet de recherche *Digital Violin Tutor* (DVT) [85] décompose le module du spectre fréquentiel en plusieurs segments adjacents d'indice n d'une longueur en fréquence de 100 cents, et ce, centrés sur la fréquence fondamentale de chacune des notes d'indice n . Ensuite, il calcule une force $A[n]$ selon l'équation 2.6 où $Z[n]$ correspond à la somme de toutes les amplitudes contenues dans le segment n . Les valeurs de k , s'expliquent par le fait que la fréquence fondamentale et les cinq premières harmoniques d'une note se situent dans les segments n , $n + 12$, $n + 19$, $n + 24$ et $n + 28$ respectivement. Finalement, les notes jouées correspondent aux valeurs maximales de $A[n]$ supérieures à un seuil ajusté expérimentalement. Il est à noter que c'est une version plus récente de cet algorithme qui a été implémentée pour les résultats de la figure 4.9 de la section 4.4, c'est-à-dire l'algorithme décrit par la référence [41]. Le principal inconvénient de ces algorithmes est que les segments n peuvent parfois contenir plus d'une harmonique lorsque plus d'une note est jouée en simultanée et cela peut parfois conduire à de mauvaises estimations de tonies.

$$A[n] = \sum_{k=0,12,19,24,28} Z[n+k] \quad (2.6)$$

2. À titre informatif, l'annexe C présente, de façon détaillé, la relation entre les notes de musique et le domaine de fréquence.

3. Le cent est une mesure liée de la théorie musicale, elle est utilisée pour calculer la distance entre deux tonies selon une échelle de fréquence logarithmique.

4. En musique, des notes séparées entre elles par une distance de 100 cents sont dites accordées selon un tempérament égal.

Pour certains instruments à cordes tels que le piano et la guitare, la fréquence des harmoniques n'est pas exactement égale à un multiple de la fréquence fondamentale. La position des harmoniques est parfois mieux estimée par l'équation 2.7 où la variable β représente le facteur d'inharmonicité [35].

$$f_n = n \cdot f_0 \sqrt{1 + \beta(n^2 - 1)} \quad (2.7)$$

Ce phénomène, nommé inharmonicité, est proportionnel à la rigidité d'une corde, c'est-à-dire que plus une corde est rigide, plus le facteur d'inharmonicité sera élevé [19]. Néanmoins, selon des observations passées avec des signaux de violon, ce phénomène semble assez négligeable pour les premières harmoniques. Ceci peut s'expliquer par le fait que le frottement constant de l'archet avec la corde permet de garder la périodicité du signal relativement constante. Ainsi, pour le violon, bien que la fréquence fondamentale réelle d'une note jouée ne soit pas toujours égale à la fréquence fondamentale nominale attendue, la fréquence des premières harmoniques est généralement égale à un multiple de la fréquence fondamentale. Par conséquent, l'AETM développé pour ce projet de doctorat s'inspire d'un AETM conçu pour les signaux harmoniques avec peu d'inharmonicité, c'est-à-dire l'AETM présenté par l'article [27] et décrit par l'équation 2.8 [6].

$$p = \arg \max_{f < f_{max}} \int_0^\infty |X(f')| \sum_{k=1}^n 0.84^{k-1} \delta(f' - kf) df' \quad (2.8)$$

L'équation 2.8 permet de trouver la fréquence f du spectre $X(f)$ pour laquelle la somme de l'amplitude de cette fréquence avec l'amplitude pondérée par un facteur 0.84^{k-1} de toutes ses harmoniques d'indice $k = 2, 3, \dots, n$ donne la plus grande amplitude. Cette fréquence correspondra ainsi à la tonie de la note jouée, soit une tonie inférieure à la tonie maximale pouvant être générée par l'instrument de musique (f_{max}). En polyphonie, il s'agirait simplement de sélectionner les N fréquences donnant les plus grandes sommes de l'intégrale. Le principal inconvénient de cette méthode est que le nombre de notes jouées en simultanément doit être connu, puisque cette fonction ne vérifie pas si l'énergie de la note est suffisante. Il serait probablement préférable de faire la sélection en recherchant les fréquences f où l'énergie de la somme de l'intégrale est supérieure à un seuil fixé. D'ailleurs, l'algorithme utilisé pour ce projet de doctorat utilise plusieurs critères basés sur l'observation des signaux de violon pour sélectionner les meilleurs candidats à partir des meilleurs résultats de sommes d'intégrales d'une équation similaire à l'équation 2.8.

2.1.3 Algorithmes basés sur la psychoacoustique

Comme le musicien est très efficace pour trouver les notes contenues dans un signal polyphonique, certains chercheurs tentent de reproduire le fonctionnement physique et psychologique d'un musicien pour estimer les différentes tonies contenues dans un signal polyphonique. Ces recherches sont liées au domaine de la psychoacoustique, c'est-à-dire l'étude de la perception auditive humaine, plus spécifiquement, sur la physiologie de l'oreille et sur le fonctionnement du cerveau. Ces algorithmes sont intéressants, car si le modèle est une bonne approximation du fonctionnement physique et psychologique d'un musicien, la fréquence fondamentale obtenue sera la même que celle perçue par les musiciens en général.

Il a été démontré par Fletcher [18] qu'une partie du fonctionnement de la cochlée peut être modélisée par un banc de filtres passe-bande dont la largeur de la bande de chacun des filtres augmente de façon logarithmique, ces filtres sont nommés *auditory filters* en anglais [26]. Ainsi, certains auteurs [32, 36, 75] décomposent d'abord le signal numérique en plusieurs sous-bandes similaires à celles retrouvées dans la cochlée pour ensuite faire d'autres traitements sur la sortie des filtres passe-bande.

Ensuite, il a été démontré que les cellules ciliées (*hair cells*) situées à l'intérieur de la cochlée permettent de transformer en impulsions nerveuses les pressions mécaniques produites par les sons. En traitement du signal, une partie du fonctionnement de ces cellules peut être modélisée par un redressement du signal temporel suivi d'un filtrage passe-bas [32, 46]. Ces deux opérations permettent de faire ressortir la périodicité de l'enveloppe temporelle.

D'abord, le redressement est un opérateur non linéaire qui a pour conséquence d'amplifier les premières harmoniques d'un signal périodique. Ensuite, le filtre passe-bas permet de retirer les harmoniques d'ordre élevé afin de conserver que les premières harmoniques. Lorsque la fréquence de coupure du filtre passe-bas est bien ajustée, le signal résultant de ces deux opérations est un signal dont la période correspond à celle de l'enveloppe temporelle.

Il a été démontré que lorsqu'un son est composé d'au moins une harmonique, la période résultant des deux opérations correspond généralement à la fréquence fondamentale puisque les harmoniques s'amplifient et s'annulent alternativement [35]. Ainsi, il est possible d'estimer les fréquences fondamentales contenues dans un signal polyphonique en trouvant d'abord cette période pour chacun des filtres passe-bande modélisant le fonctionnement de la cochlée et en analysant l'ensemble de ces périodes pour estimer les notes contenues dans le signal.

2.2 Détections des transitions

En traitement audio, une transition correspond à un segment audio représentant les premiers instants de la formation d'un nouvel événement sonore. Tout comme pour les algorithmes d'estimation de la fréquence fondamentale, les algorithmes de détection de transitions peuvent être basés sur un ou plusieurs domaines tels que le domaine temporel et fréquentiel. Ces algorithmes sont essentiels pour le système proposé puisqu'ils permettent de détecter le début et la fin des notes pour éventuellement leur associer une durée dans la symbolique musicale telle qu'une noire, une blanche ou une croche.

2.2.1 Algorithmes basés sur les paramètres temporels

Certaines transitions sont très apparentes en observant le signal temporel. Par exemple, les premières millisecondes du signal temporel d'une note jouée au piano possèdent toujours des amplitudes plus élevées, car le début de la note est produit par le frappement d'un marteau sur la corde. C'est pourquoi certains algorithmes utilisent la forme de l'enveloppe temporelle [71] ou la forme de l'énergie du signal audio [16] pour détecter les débuts de notes. Il est à noter que pour diminuer le temps de traitement pour l'analyse de l'enveloppe, ces algorithmes utilisent généralement une version sous échantillonnée du signal temporel. De plus, ces algorithmes sont généralement plus performants lorsqu'un prétraitement est utilisé pour accentuer les paramètres temporels d'intérêts tels qu'une décomposition en sous-bandes [34].

2.2.2 Algorithmes basés sur les paramètres fréquentiels

Il a été démontré que les segments audio correspondant aux attaques des notes possèdent généralement plus d'énergie dans les hautes fréquences [68]. Ainsi, il est possible de détecter les débuts de notes en sommant ensemble les amplitudes des hautes fréquences contenues dans le module du spectre fréquentiel d'une trame et, ensuite, en comparant le résultat à un seuil. Pour obtenir de meilleurs résultats, les fréquences peuvent d'abord être pondérées pour avantager certaines fréquences par rapport à d'autres.

Comme les amplitudes du module du spectre fréquentiel dépendent de l'énergie du signal, comparer à un seuil le résultat de la sommation des amplitudes pondérées peut causer des fausses détections en présence de sons de très forte énergie ou de sons contenant plusieurs notes jouées simultanément. De plus, les attaques jouées en douceur risquent de ne pas produire suffisamment d'énergie en haute fréquence pour causer le dépassement du seuil. Ainsi, certains chercheurs tentent plutôt de trouver des variations brusques d'énergie en comparant les résultats de deux trames adjacentes [16, 39].

Les algorithmes utilisant le module du spectre sont généralement plus performants en présence de signaux audio polyphoniques que les algorithmes basés sur des paramètres temporels. Par contre, ils sont peu performant en présence d'attaques très douces telles que celles qui peuvent être produites par un violon.

Tout comme le module, les phases $\tilde{\varphi}(m, k)$ du spectre peuvent être utilisées pour détecter les transitions où m et k correspondent respectivement à l'indice de la trame et à l'indice de la phase dans le spectre fréquentiel. En effet, la déviation de phase $(\tilde{\varphi}(m, k) - \tilde{\varphi}(m - 1, k))$ de la majorité des raies spectrales k devrait rester sensiblement constante d'une trame m à l'autre lorsque le signal audio est périodique. Ainsi, l'angle différentiel $d\tilde{\varphi}$ d'une raie spectrale k défini selon l'équation 2.9 [3] devrait se rapprocher de la valeur zéro pour un signal périodique. Il est à noter que la fonction $\text{princarg}[x]$ permet d'obtenir l'argument principal de x , et ce, dans l'intervalle $]-\pi, \pi]$.

$$d\tilde{\varphi}(m, k) = \text{princarg}[\tilde{\varphi}(m, k) - 2\tilde{\varphi}(m - 1, k) + \tilde{\varphi}(m - 2, k)] \quad (2.9)$$

Inversement, en présence de signaux moins périodiques, tels que pendant la période transitoire, la moyenne des valeurs absolues de tous les angles différentiels du spectre devrait être élevée [3]. Ainsi, il est possible d'appliquer un seuil sur cette moyenne pour détecter les transitions.

Contrairement aux algorithmes basés sur le module du spectre, le seuil utilisé est moins sensible à l'énergie du signal puisque la phase n'est pas reliée à l'énergie du signal. Par conséquent, ces algorithmes sont plus efficaces pour détecter des attaques très douces telles que celles qui peuvent être produites par un violon. Par contre, comme l'énergie du signal n'est pas considérée, ces algorithmes sont plus sensibles au bruit. Comme le bruit n'est pas nécessairement périodique, la déviation de phase n'est pas nécessairement constante d'une trame à l'autre, provoquant ainsi de fausses détections. D'ailleurs, comme les avantages des algorithmes basés sur la phase et le module sont complémentaires, certains algorithmes utilisent une combinaison des deux paramètres pour détecter les transitions [15].

2.2.3 Paramètres temporels et fréquentiels jumelés

Tout comme la transformée de Fourier qui décompose le signal temporel en plusieurs sinus et cosinus de fréquence multiples, la décomposition en ondelettes décompose le signal temporel en une fonction de base de dimensions multiples telle que la fonction de base de Daubechies [13].

Contrairement aux fonctions sinus et cosinus qui possèdent un support infini et qui, par conséquent, assument que le signal est périodique, les fonctions des ondelettes sont localisées dans le temps. D'ailleurs, plus le support de la fonction de base d'une ondelette est petite, plus la résolution temporelle est élevée et plus le support de la fonction de base est grand, plus la résolution fréquentielle est élevée. Ainsi, contrairement à la transformée de Fourier où les résolutions temporelles et fréquentielles sont constantes, la décomposition en ondelettes permet de faire des compromis entre une bonne résolution temporelle et fréquentielle. Cet aspect est un avantage important pour détecter des transitions, car pour observer les changements rapides, comme à la transition d'une note à l'autre, il faut une bonne résolution temporelle et pour observer les changements de fréquences, il faut une bonne résolution fréquentielle.

Pour toutes ces raisons, beaucoup d'algorithmes utilisent la décomposition en ondelettes pour détecter les transitions. Par exemple, certains chercheurs [80] utilisent la décomposition en ondelettes discrète et d'autres [78] utilisent la décomposition en ondelettes harmonique [52] pour détecter des débuts de notes. La décomposition en ondelette peut s'avérer intéressante pour détecter les transitions, par contre, dans plusieurs cas, elle s'avère trop complexe pour être utilisée dans des applications en temps réel.

CHAPITRE 3

ÉTAT DE L'ART - MÉTHODES D'ALIGNEMENT

Bien que les algorithmes de traitement du signal discutés dans le chapitre précédent permettent de récupérer de l'information pertinente à l'estimation des notes contenues dans un signal audio numérique, cette information doit être traitée par d'autres algorithmes pour permettre la réalisation d'un système informatique tel que défini par la question de recherche de ce projet de doctorat.

En effet, cette information a d'abord besoin d'être traitée pour permettre le suivi de la prestation musicale du musicien à travers la partition de musique. Il est à noter que ces traitements sont normalement réalisés à l'aide de méthodes d'alignement et que ces méthodes doivent tenir compte de plusieurs facteurs pour être optimales. D'abord, ces méthodes doivent tenir compte du fait qu'aucun algorithme d'estimation de tonies multiples et d'estimation de transition n'est parfait, il y a généralement un nombre non négligeable de tonies et transitions manquantes ou estimées en trop. Ensuite, ces méthodes doivent tenir compte du fait que le musicien peut oublier ou jouer des notes en trop et peut aussi commencer à jouer des notes trop tôt ou trop tard. Toutes ces déviations par rapport à la partition de musique ont pour effet de complexifier le suivi de la partition de musique, résultant ainsi en la nécessité de réaliser des méthodes d'alignement très complexes.

Le premier algorithme d'alignement du signal audio à une partition musicale a été publié en 1984 [11]. Le système proposé utilisait une fenêtre glissante à faible portée permettant de comparer les notes jouées avec les notes de la partition.

De nos jours, il existe principalement deux classes de méthodes d'alignement de la partition. La plus ancienne est la déformation dynamique du temps (*Dynamic Time Warping*, DTW) [2, 12, 28, 69, 70, 73, 76] et la plus récente est basée sur les modèles de Markov cachés (*Hidden Markov Model*, HMM) [8, 51, 54, 63, 72]. Il existe aussi d'autres techniques très intéressantes basées sur les modèles bayésiens [43, 64, 65]. Les sections suivantes présentent les plus importantes de ces techniques.

3.1 Méthodes basées sur les HMM

Les méthodes d'alignement basées sur les HMM sont les modèles statistiques les plus utilisés pour effectuer le suivi de la partition. En entraînant d'abord ces modèles avec les

performances musicales d'un musicien, et ce, pour une partition de musique donnée, ces modèles peuvent s'adapter au jeu du musicien à l'aide de ces apprentissages. De plus, il est possible de les utiliser en temps réel avec des techniques de programmation dynamique. Afin de faciliter la lecture, un court rappel sur les HMM ainsi qu'un exemple adapté au contenu de cette section sont disponibles à l'annexe A.

Afin de pouvoir aligner les notes jouées avec les notes de la partition, les notes contenues dans la partition sont d'abord modélisées par un HMM. Le modèle utilisé pour représenter la note est généralement constitué de plusieurs états appartenant à trois classes d'états possibles : l'attaque, le maintien de la note que l'on nomme *sustain* en anglais et le silence. La figure 3.1 illustre un exemple d'un HMM pour une note.

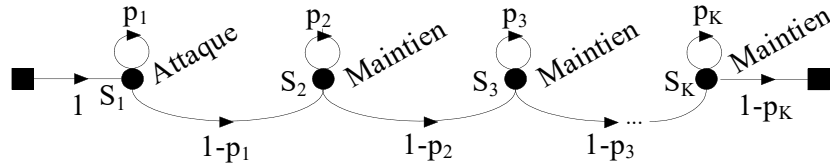


Figure 3.1 Exemple d'un HMM utilisé modélisant une note

Le nombre K d'états consécutifs dans une même classe et la probabilité de transition p d'un état vers lui-même pour tous les états d'une classe donnée sont choisis dans le but d'obtenir une forme désirée de distribution de probabilité. Par exemple, les chercheurs de la référence [54] utilisent, pour déterminer les valeurs de p et de K associées à une même classe de maintien, la distribution de Pascal (*negative binomial distribution*) où l'abscisse représente le nombre n de trames associé à la classe et l'ordonnée représente la probabilité que le nombre n corresponde au nombre N de trames attendu pour cette classe. En effet, la distribution de Pascal représente une probabilité de succès par rapport à un nombre discret, soit, dans ce cas, la probabilité que n corresponde à N , c'est-à-dire $P(N = n)$. Pour cet exemple, $P(N = n)$ est défini à l'aide de l'équation 3.1 où la forme $\binom{a}{b}$ correspond à un coefficient binomial.

$$P(N = n) = \binom{n-1}{K-1} p^{n-K} (1-p)^K \quad (3.1)$$

Certains [63] suggèrent d'utiliser le modèle de la figure 3.2 en présence de notes de courte durée où le nombre attendu de trames associé à la classe maintien serait faible. Dans ce modèle, chacun des états correspond à une trame. Ainsi, la probabilité p_k correspond à la probabilité que la note possède une durée en trame plus élevée que $k+1$ trames : $p_k = P(T > k+1 | T > k)$ où T correspond au nombre de trames dans la note.

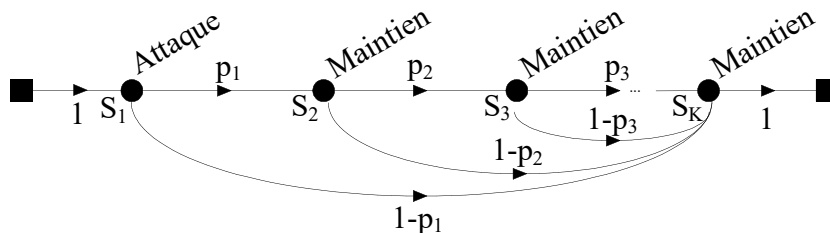


Figure 3.2 HMM plus performant pour une note de courte durée

Ce modèle est très efficace pour des notes de courte durée. Par contre, lorsque les notes sont de longue durée, le modèle contient beaucoup d'états et le calcul de la séquence la plus probable devient beaucoup plus coûteux.

Il arrive parfois que le musicien ne joue pas la note sur toute sa durée en ajoutant un effet tel que le staccato. C'est pourquoi certains chercheurs [54, 72] ajoutent un état facultatif de silence à la fin de la séquence d'état tel qu'illustré à la figure 3.3.

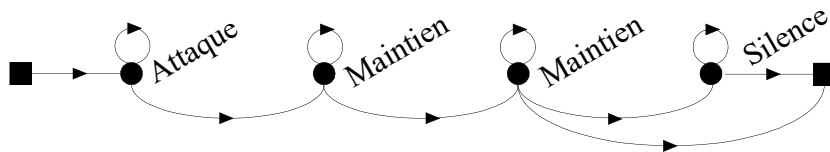


Figure 3.3 HMM permettant de modéliser les notes jouées de différentes façons, par exemple Staccato

Le problème avec le modèle de la figure 3.3 est que le ratio entre la durée jouée (attaque et maintien) et la durée du silence est fixe. Par exemple, il est possible qu'un musicien décide de faire sonner une note que pendant la moitié de sa durée nominale tandis que le modèle a été conçu en assumant que cette même note aurait été maintenue pendant la majorité de sa durée nominale. En fait, la durée du silence ne peut être fixée pour tous les musiciens, car elle dépend de l'interprétation du musicien. Ainsi, certains auteurs utilisent plutôt des silences en parallèle [63, 65]. Par contre, ces modèles demandent beaucoup plus de traitement que le modèle avec un seul état de silence, car son nombre d'états est supérieur et ainsi la séquence optimale est plus complexe à calculer.

La partition de musique peut aussi être modélisée par un HMM. Par exemple, les chercheurs de la référence [63, 65] et de la référence [8] modélisent la partition de musique par ses N notes à l'aide d'un modèle de Markov caché discret tel qu'illustré à la figure 3.4.

Le problème avec ce modèle est qu'il estime que le musicien joue exactement le bon nombre de notes et les bonnes notes, ce qui n'est pas toujours le cas. En effet, le musicien peut oublier de jouer des notes ou jouer des notes en trop. Certains corrigent le problème en

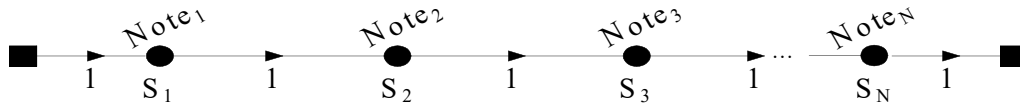


Figure 3.4 Modélisation de la partition note par note

ajoutant des états fantômes en parallèle avec chacune des notes tel que démontré à la figure 3.5 [54, 72].

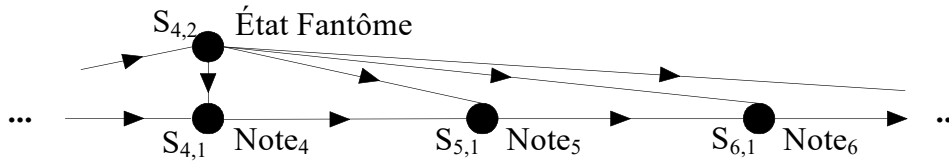


Figure 3.5 Modélisation de la partition avec des notes et des états fantômes

Ce modèle permet de modéliser les notes manquantes, les notes en trop et les notes erronées. Par exemple, une note jouée en trop passerait d'abord par l'état $S_{4,2}$ et ensuite par l'état $S_{4,1}$, une note manquante passerait d'abord par l'état $S_{4,2}$ et ensuite par l'état $S_{6,1}$ et une note erronée passerait par l'état $S_{4,2}$ et ensuite par l'état $S_{5,1}$.

Afin de pouvoir bien profiter du modèle HMM utilisé pour représenter les différentes phases de progression d'une note telles que la phase d'attaque, de maintien et de relâchement (silence), il est nécessaire d'extraire des paramètres audio susceptibles de bien refléter ces états. Par exemple, la fréquence fondamentale pourrait être utilisée pour représenter le maintien d'une note monophonique.

Généralement, les paramètres observables sont choisis en fonction de l'instrument de musique à modéliser. Par exemple, certains auteurs [72] modélisent une note de piano par l'évolution de son énergie dans le temps, comme le montre la partie gauche de la figure 3.6.

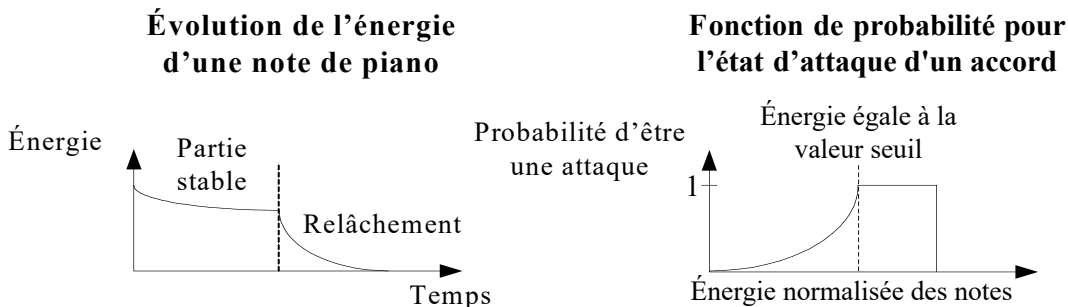


Figure 3.6 Exemple de paramètres observables pour des notes de piano

Effectivement, durant le maintien d'une note de piano, l'énergie décroît lentement et lors du relâchement de la note, l'énergie décroît plus rapidement. Pour l'état d'attaque d'un accord¹ donné, le modèle compare l'énergie normalisée de l'ensemble des notes jouées à un instant donné avec une valeur seuil d'énergie où au-delà de cette valeur, la probabilité d'être l'attaque attendue est de 1, telle qu'illustrée à la figure 3.6 à droite.

La forme de la fonction de la figure 3.6 à droite s'explique comme suit. Durant l'attaque, toutes les notes à jouer ne sont pas nécessairement présentes, car parfois les notes sont jouées en arpège². La valeur seuil détermine un niveau d'énergie acceptable en considérant que certaines des notes pourraient ne pas être encore présentes au moment de la mesure de l'énergie.

Le principe de comparaison entre l'énergie obtenue et l'énergie attendue est aussi utilisé pour modéliser les états de maintien et l'état de relâchement (état *rest* en anglais), mais une fonction de probabilité de forme gaussienne est utilisée puisque normalement toutes les notes devraient être présentes à ces états.

Afin d'éviter de prendre des paramètres observables non optimaux, les chercheurs de la référence [8] utilisent plus d'un paramètre observable par état tel que la différence d'énergie, le taux de passage par zéro et la fréquence fondamentale. Ensuite, ils appliquent une quantification vectorielle sur le vecteur d'observation afin d'obtenir un seul indice représentant tous les paramètres observables. Finalement, ils construisent une densité de probabilité gaussienne à partir des indices. Cette méthode est très efficace, mais comme les paramètres observables sont quantifiés en une seule valeur, il y a des pertes d'information dues à l'erreur de quantification. Ces pertes ont pour effet de diminuer la performance pouvant être atteintes par de ces méthodes.

Finalement, certains algorithmes d'alignement pour les musiques chantées [7, 40] utilisent des paramètres observables de la reconnaissance vocale. Par exemple, ils peuvent reconnaître les phonèmes prononcés par la personne et ainsi améliorer les performances de l'alignement puisque les phonèmes sont généralement associés à une note dans la partition.

1. Un accord correspond à un ensemble de notes normalement jouées en simultanément.

2. Un arpège est un ensemble de notes d'un même accord où ces dernières sont jouées successivement et avec rapidité pour, ensuite, être maintenues jusqu'à la fin de la durée nominale de l'accord.

3.2 Modèles bayésiens

La majorité des techniques de HMM font l'hypothèse d'un tempo³ fixe pour construire le modèle de la partition et ensuite entraînent les probabilités de transition à partir de plusieurs interprétations de la pièce. En réalité, le tempo peut changer sensiblement d'un musicien à l'autre. Certains musiciens peuvent jouer une pièce deux fois plus rapidement que d'autres. De plus, lors de l'apprentissage, les musiciens n'arrivent généralement pas à maintenir le tempo d'une façon rigoureuse.

Les modèles bayésiens permettent de représenter graphiquement les dépendances entre différentes variables aléatoires. Par exemple, certains auteurs proposent de modéliser le tempo par le modèle bayésien de la figure 3.7 [64], et ce, afin de pouvoir générer les notes d'une musique d'accompagnement aux bons moments. Pour cet exemple, les variables aléatoires sont représentées par des cercles où les cercles gris et blanc représentent des variables aléatoires observables et non observables respectivement.

Modèle Bayésien	Définition des variables aléatoires
	a_n : Début (en s.) estimé d'une note n , et ce, à l'aide d'un algorithme de détection de transitions utilisant le signal audio de la partie soliste
	τ_n : Déviation (en s.) du début de la note n par rapport au début nominal σ_n : Déviation du tempo (en s./mesure) au moment où la note n est jouée par rapport au tempo nominal
	t_n : Temps (en s.) correspondant au début de la note s_n : Tempo local (en s./mesure) au moment où la note n est jouée
	b_n : Début estimé d'une note n , et ce, à l'aide d'un algorithme de détection de transitions utilisant le signal audio de la partie d'accompagnement
● : Variables aléatoires observables ○ : Variables aléatoires cachées	

Figure 3.7 Modèle bayésien permettant de modéliser le tempo

Avant d'utiliser le modèle de la figure 3.7, les fonctions de densité de probabilité associées aux variables aléatoires τ_n , σ_n (voir figure 3.7) sont d'abord déterminées à l'aide d'entraînements et à l'aide de l'équation 3.2 où la variable l_n correspond à la distance en mesure entre la note n et $n+1$. Les entraînements utilisent comme entrée les débuts estimés a_n des notes de la partie soliste et les débuts estimés b_n des notes de la partie d'accompagnement

3. Le tempo défini la vitesse d'exécution d'une pièce de musique. Par défaut, son unité de mesure correspond au nombre de battements par minute.

jouée par un musicien, et ce, obtenus à partir d'un algorithme de détection de transitions basé sur des HMM. Une fois entraîné, le système estime de nouveau le moment de début a_n des notes jouées par le soliste, mais cette fois en l'absence de la partie d'accompagnement. À partir de chacune des valeurs a_n obtenues, le module calcule le moment de début t_{n+1} de la prochaine note d'accompagnement b_{n+1} à générer et le tempo s_{n+1} à utiliser pour le prochain calcul, et ce, toujours à l'aide de l'équation 3.2 où t_n est estimée à partir de la valeur de a_n . Il est à noter que le tempo local s_n à utiliser pour ces prédictions correspond au tempo ayant la probabilité maximale de la fonction de densité de probabilité correspondante.

$$\begin{pmatrix} t_{n+1} \\ s_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & l_n \\ 0 & 1 \end{pmatrix} \begin{pmatrix} t_n \\ s_n \end{pmatrix} + \begin{pmatrix} \tau_n \\ \sigma_n \end{pmatrix} \quad (3.2)$$

L'avantage de cette méthode est que le tempo local s_n est constamment réestimé en considérant le tempo de toutes les notes précédentes et les déviations du tempo apprises lors des performances précédentes. Effectivement, il est possible d'extraire les intentions rythmiques du musicien en observant les moyennes et les variances des distributions τ_n et σ_n . Par exemple, lorsque la moyenne et la variance de la distribution σ_n sont négatives et nulles respectivement, ceci indique que le musicien diminue de façon systématique le tempo à ce moment donné, et ce, pour chacune de ses interprétations de la partition. Par contre, le problème est que certaines des déviations apprises peuvent être dues à des erreurs de tempo systématiques répétées par un musicien non expérimenté.

L'un des grands désavantages des modèles bayésiens développés pour le suivi de la partition de musique est qu'ils ne sont pas conçus pour être robustes en présence d'erreurs telles que des notes en trop ou des notes manquantes.

3.3 Déformation dynamique du temps

L'algorithme de la déformation dynamique du temps (DTW - *Dynamic Time Warping*) est généralement utilisé pour aligner deux signaux de même nature, tels que deux signaux audio numériques. Les signaux à aligner sont habituellement de forme similaire, mais dont les dimensions peuvent être étirées ou écrasées. Pour trouver l'alignement optimal, la méthode DTW utilise une matrice de similarité où chacun des éléments représente une mesure de similarité entre deux éléments distincts des signaux.

La figure 3.8 illustre un exemple où deux signaux ayant des dimensions différentes ont été alignés à l'aide de l'algorithme DTW. La partie gauche de la figure correspond à une

matrice de similarité où l'intensité des éléments est proportionnelle à la similarité entre les deux éléments comparés du *Signal 1* et du *Signal 2*. Par exemple, l'échantillon 0 du *Signal 2* est très similaire aux échantillons aux alentours des échantillons 0 et 20 du *Signal 1*.

L'algorithme DTW utilise la matrice de similarité pour aligner chacun des échantillons du *Signal 1* avec un échantillon unique du *Signal 2* tel qu'illustré à droite de la figure 3.8. Afin de faire l'alignement des deux signaux, l'algorithme DTW fait d'abord l'hypothèse que le premier échantillon du *Signal 1* correspond au premier échantillon du *Signal 2* (point A de la figure 3.8) et que le dernier échantillon du *Signal 1* correspond au dernier échantillon du *Signal 2* (point B de la figure 3.8). Ensuite, l'algorithme recherche la plus petite distance pour joindre le point A avec le point B en passant par différents points de la matrice de similarité. La valeur d'une distance correspond à la sommation des éléments pondérés de la matrice de similarité faisant partis du chemin utilisé pour se rendre du point A au point B. Il est à noter que si les deux signaux comparés sont identiques, l'alignement obtenu sera une diagonale parfaite.

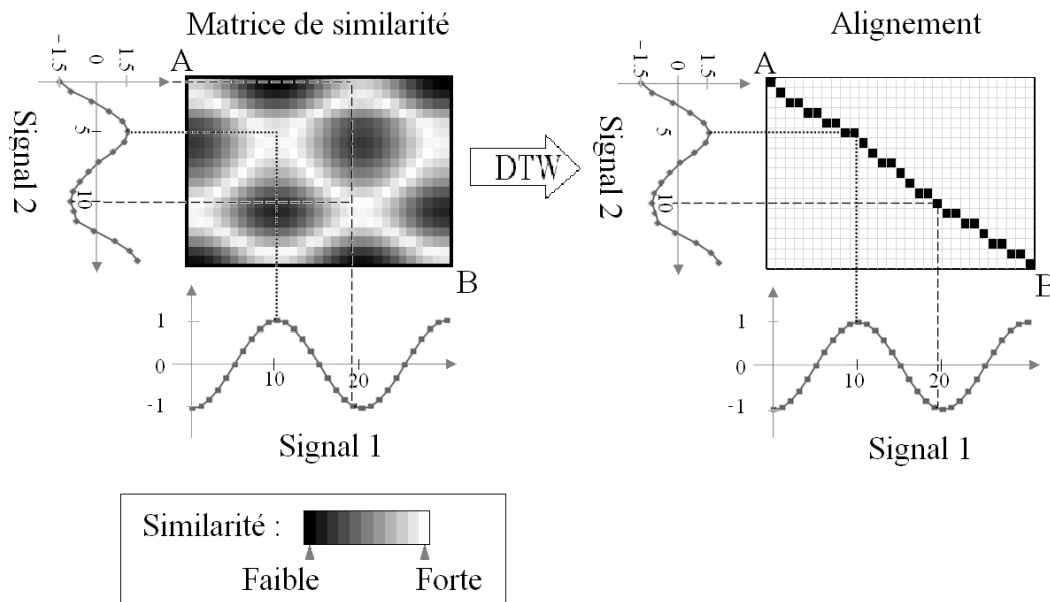


Figure 3.8 Exemple d'alignement obtenu par le DTW.

Les méthodes d'alignement basées sur le DTW sont les plus anciennes, mais sont encore utilisées aujourd'hui. Elles sont très simples à réaliser et ne nécessitent aucun entraînement. Par contre, la recherche de l'alignement optimal demande beaucoup de traitement et peut ainsi ralentir considérablement le système. C'est pour cette raison que ces techniques sont généralement utilisées pour faire de l'alignement en différé.

Normalement, l'alignement de la partition de musique avec un signal audio réel par DTW est effectué en quatre étapes :

Étape 1 : Transformation du fichier MIDI en fichier audio⁴

Étape 2 : Extraction des paramètres audio d'intérêts

Étape 3 : Calcul des distances locales entre les signaux de la partition et la performance

Étape 4 : Recherche du chemin d'alignement optimal pour minimiser la distance globale

Les chercheurs des références [2, 12, 28] transforment les signaux audio sous forme de chromagramme discret, c'est-à-dire un chromagramme par trame. Le chromagramme est un vecteur composé de 12 éléments correspondant aux douze notes de musique (*Do*, *Do#*, ..., *Si*). Les valeurs de chacun des éléments du vecteur correspondent à la moyenne des amplitudes des raies spectrales appartenant à ces notes. Pour ces techniques, les éléments de la matrice de similarité du DTW correspondent à la distance euclidienne entre le chromagramme d'une trame x du signal audio de la partition (MIDI) et le chromagramme d'une trame y du signal audio correspondant à la partition de musique jouée par le musicien.

L'avantage des chromagrammes est qu'ils peuvent compacter l'information harmonique du signal en seulement 12 éléments correspondant aux 12 notes de musiques, et ce, indépendamment de l'octave. Ainsi, les chromagrammes ne sont pas sensibles aux particularités du spectre tel que les formants puisque les particularités sont toutes regroupées dans l'un des 12 éléments. C'est pour cette raison que ces techniques peuvent utiliser des signaux audio synthétisés (MIDI) et les comparer avec des signaux réels. Les chromagramme sont sensibles aux notes prédominantes, car celles-ci accentuent l'un des 12 éléments du chromagramme. Ainsi, ces techniques peuvent fonctionner en présence de signal très polyphonique avec des mélodies qui produisent des notes prédominantes telles que l'on retrouve souvent dans les musiques populaires. L'un des désavantages des chromagrammes est que les notes sont toutes classifiées dans une seule octave et ainsi, une partie des subtilités de la pièce est perdue.

Ces méthodes considèrent seulement les fréquences fondamentales et ne considèrent pas d'autres paramètres importants pouvant renforcer l'alignement tel que la détection de début de période transitoire (*onsets detection*). La distance de la structure des raies spectrales (*Peak Structure Distance*, PSD) [73, 76] est une méthode considérant à la fois les fréquences fondamentales et les débuts de période transitoire. Elle consiste à trouver la

4. Pour réduire la quantité de traitement, il est possible d'extraire directement les événements temporels du format MIDI au lieu de passer par le convertisseur MIDI/audio.

différence (distance) entre les modules du spectre P_i d'une trame m réelle (performance du musicien) et les modules du spectre S_i de la trame n du signal attendu (partition) selon l'équation 3.3 où l'indice i correspond aux raies spectrales. Le spectre du signal attendu est construit à partir du signal MIDI et le maximum de toutes ces harmoniques est fixé à 1 pour obtenir des valeurs de PSD entre 0 et 1 inclusivement. La figure 3.9 illustre un exemple de spectre attendu construit à partir du signal MIDI. Selon l'équation 3.3, lorsque le spectre P d'une trame réelle ressemble au spectre attendu S , la valeur PSD calculée sera près de zéro et lorsque la ressemblance sera faible, la valeur PSD calculée sera près de 1. En effet, pour obtenir une faible valeur PSD, l'énergie du spectre de la trame réelle doit être forte dans les zones à l'intérieur des pointillés et elle doit être faible dans les zones en dehors des pointillés de la figure 3.9 .

$$PSD(m, n) = 1 - \frac{\sum_i S_i P_i^2}{\sum_i P_i^2} \quad (3.3)$$

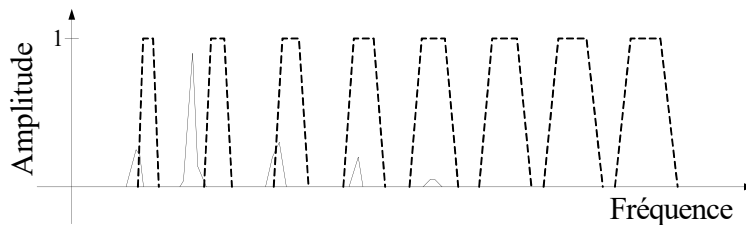


Figure 3.9 Exemple d'un spectre attendu S_i (trait pointillé) et réel P_i (trait continu)

Le PSD est efficace pour des signaux monophoniques. Par contre, en présence de signaux polyphoniques, les notes de courte durée et de faibles intensités sont difficiles à détecter par le PSD puisque les notes concurrentielles tendent à recouvrir la majeure partie du spectre et ainsi donner l'impression que les notes de courte durée de faible intensité sont présentes en tout temps. Idéalement, les bandes fréquentielles du spectre désiré (partition) doivent être sélectives (étroites) pour éviter de prendre en compte l'énergie reliée à d'autres notes. Par contre, les filtres à bandes étroites ne fonctionnent pas pour des instruments dont la fréquence fondamentale peut se déplacer, comme le violon. Effectivement, des effets tels que le vibrato ou de fausses notes peuvent facilement sortir de la bande étroite. Il est à noter que certains auteurs [76] utilisent des bandes d'une forme gaussienne au lieu d'une forme trapézoïdale pour donner moins d'importance aux fréquences éloignées de la bande idéale.

Les techniques basées sur le PSD utilisent aussi des distances liées au début des notes (*onset detection*) afin d'améliorer la précision de l'alignement. Certains auteurs [73] utilisent

simplement la différence entre le PSD actuel et le PSD précédent selon l'équation 3.4 pour détecter les débuts de note.

$$\Delta PSD(m, n) = \begin{cases} \delta - \theta_d & \text{si } \delta \geq \theta_d \\ 0 & \text{si } \delta < \theta_d \end{cases} \quad (3.4)$$

où θ_d correspond à une valeur seuil trouvée expérimentalement et

$$\delta = PSD(m, n) - PSD(m - 1, n)$$

Ce type de distance ne fonctionne pas bien en présence de notes répétitives, par exemple Do, Do, Do, etc., puisqu'elle est basée sur la structure du spectre de la note et les notes répétitives possèdent exactement la même structure. Pour cette même raison, si les harmoniques de la note sont très similaires à celles de la note précédente, l'alignement du début de note ne devrait pas être efficace avec cette distance. Pour pallier à ce problème, certains auteurs [76] utilisent plutôt une distance liée à la différence d'énergie obtenue dans chacune des bandes du spectre attendu. Ainsi, cette méthode est sensible aux différences de puissance et à la structure du spectre et peut donc détecter le début des notes répétitives. Par contre, l'attaque doit être suffisamment marquée pour produire des variations de puissance suffisantes pour la détection.

De plus, ces méthodes d'alignement ajoutent un silence à la fin de chacune des notes dans le modèle de la partition pour améliorer l'alignement en présence de notes jouées avec un effet tel que le staccato. En effet, elles ajoutent une nouvelle distance (sPSD) dans la matrice de similarité correspondant à un silence à la fin de chacune des notes. Pour ce faire, elles utilisent simplement une technique basée sur l'énergie du signal selon l'équation 3.5.

$$sPSD(m, n) = \begin{cases} e - \theta_s & \text{si } e \geq \theta_s \\ 0 & \text{si } e < \theta_s \end{cases} \quad (3.5)$$

où θ_s correspond à une valeur seuil trouvée expérimentalement et $e = \log \sum P_i^2$

Les algorithmes d'alignement basés sur le DTW exigent beaucoup de mémoire, car ils doivent essayer tous les chemins possibles pour trouver le trajet optimal. Heureusement, l'ajout de contraintes (voir l'annexe B) permet de diminuer le nombre de trajets possibles et ainsi de diminuer l'espace mémoire requis. Il est aussi possible d'utiliser un corridor diagonal pour limiter le nombre de chemins possibles puisque généralement les trajets optimaux suivent la diagonale. D'ailleurs comme le DTW est très résistant aux différences de longueur des signaux audio, le nombre de trames utilisé pour coder la partition peut

être réduit par décimation avant d'utiliser l'algorithme DTW. Finalement, il est possible de réduire davantage l'espace mémoire en ne considérant pas l'évolution des notes. En effet, souvent, seulement l'information sur le début et la fin des notes est importante et ainsi les notes de la partition peuvent être résumées à trois trames : début, partie stable et fin. Malgré tout, il arrive que des pièces de longue durée utilisent trop d'espace mémoire pour être alignées par ces techniques.

3.4 EDTW

Contrairement à la majorité des algorithmes d'alignement actuels [2, 8, 11, 12, 28, 54, 63, 64, 65, 72, 73, 76], l'algorithme d'alignement EDTW (*Extended Dynamic Time Warping*) proposé lors de la maîtrise [21] utilise pour faire l'alignement des notes trouvées à partir d'algorithmes de transcription musicale au lieu d'utiliser des paramètres extraits de chacune des trames. Ainsi, la matrice de similarité est constituée d'éléments représentant une similarité entre une note jouée j et une note de la partition i , telle que présentée à la figure 3.10 où les valeurs de similarité sont représentées selon un dégradé entre le noir (similarité nulle) et le blanc (similarité = 1) respectivement. Par conséquent, la complexité et la quantité d'espace mémoire utilisée pour rechercher l'alignement optimal sont réduits par rapport aux systèmes DTW habituels basés sur les trames puisqu'il y a moins de notes que de trames et, par conséquent, moins d'éléments dans la matrice de similarité. Il est à noter que pour la version de l'algorithme EDTW de la référence [21], chacune des valeurs de similarité est calculée à partir de deux variables représentant la précision sur le moment de l'attaque et la précision sur la durée de la note jouée j en admettant que cette dernière corresponde à la note de la partition i .

La méthode EDTW peut être résumée en deux grandes étapes. La première étape consiste à trouver à partir de la matrice de similarités tous les segments de la partition de musique qui peuvent avoir potentiellement été joués. Ces segments sont nommés des chemins survivants (CS). Pour mieux illustrer cette étape, les figures 3.11(a) et 3.11(b) présentent un exemple de tous les chemins survivants obtenus à partir de la matrice de similarités de la figure 3.10. La deuxième étape de la méthode consiste à sélectionner les CS les plus probables dans le but de créer l'alignement entre ce qui a été joué et la partition. La figure 3.11(b) présente l'alignement obtenu à partir des chemins survivants de la figure 3.11(a) où les valeurs de similarité sont représentées selon le même dégradé que pour la figure 3.10.

De façon plus détaillée, la procédure EDTW est réalisée à partir des étapes suivantes pour chacune des notes jouées j :

1. Recherche des positions Ω_x des valeurs de similarités les plus fortes et non nulles

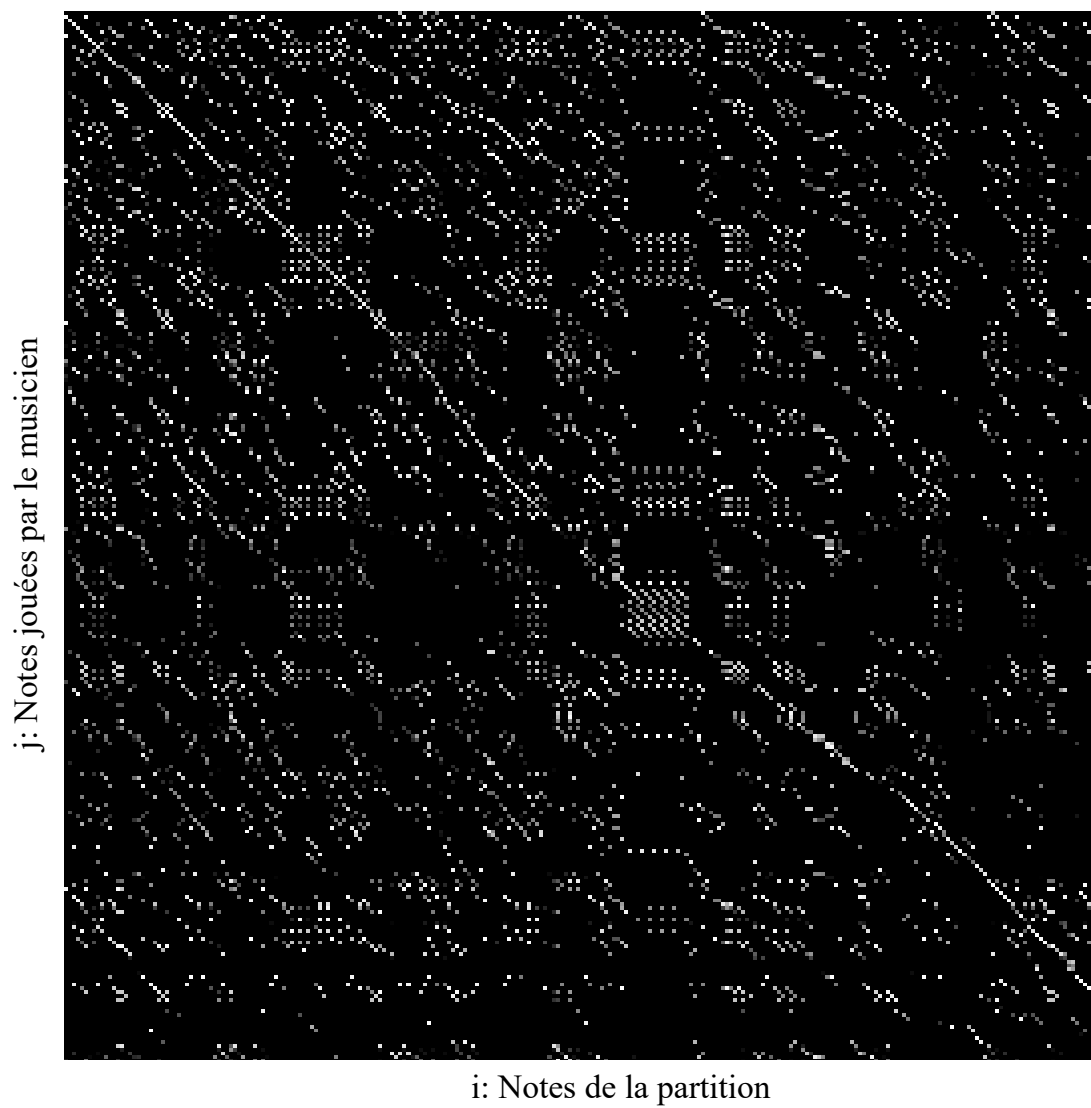


Figure 3.10 Exemple de matrice de similarités

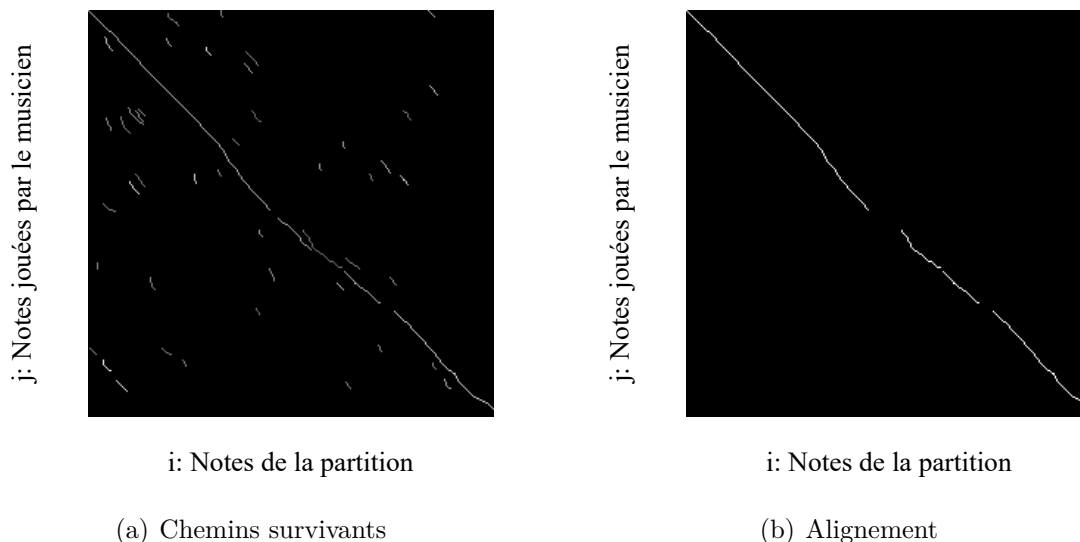


Figure 3.11 Exemple de chemins survivants et d'alignement

La première étape consiste à trouver les X positions Ω_x des valeurs de similarités les plus fortes et non nulles pour la note jouée courante j , c'est-à-dire de trouver les colonnes i de la ligne j de la matrice de similarités où les valeurs de similarités sont non nulles et parmi les X plus fortes. Il est à noter que lorsque cette étape est réalisée pour la première fois, la première note jouée correspond à la note courante j .

2. Recherche du chemin local le plus optimal (CLPO) pour chacune des positions Ω_x

Une fois les positions Ω_x trouvées, le chemin local le plus optimal (CLPO) est trouvé pour chacune de ces positions n'appartenant pas déjà à un chemin survivant. Le CLPO correspond à un alignement local potentiel en observant seulement les valeurs de similarités de la matrice de similarités à proximité de la position Ω_x . Pour ce faire, le CLPO est trouvé à l'intérieur d'un bloc de dimension $M \times N$ dont la première position correspond à une position Ω_x telle qu'illustrée à la figure 3.12 où le bloc utilisé est de dimension 3×3 .

Il existe un nombre fixe Q de chemin local (CL) possibles à l'intérieur d'un bloc $M \times N$. Chacun des CL possibles sont formés de M éléments de la matrice de similarités reliés entre eux selon les déplacements définis à la figure 3.13. La limitation des déplacements est justifiée par l'hypothèse que les notes de la partition ne seront jamais jouées à reculons dans le temps et que toutes les notes jouées peuvent seulement être associées avec une ou plusieurs notes consécutives de la partition. De plus, comme les déplacements verticaux et horizontaux devraient se produire qu'occasionnellement, c'est-à-dire que lorsque le musicien a répétée ou sautée une note de

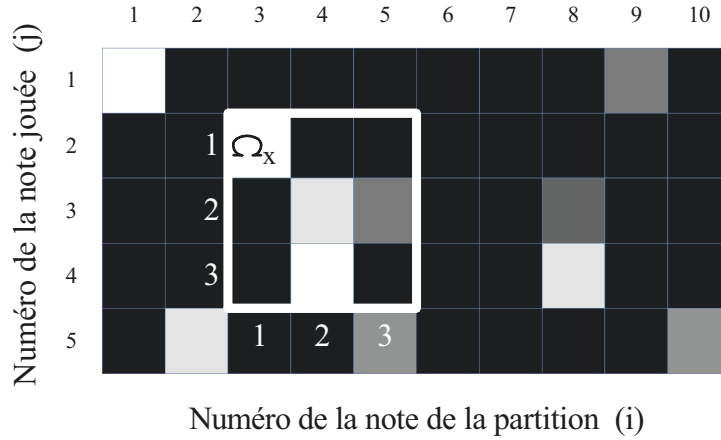


Figure 3.12 Exemple d'un bloc $M \times N$ à l'intérieur de la matrice de similarités

la partition, un coefficient $\alpha_{k,q}$ est associé à ces deux déplacements pour favoriser le déplacement diagonal tel que défini avec l'équation 3.6 où m et n représentent respectivement le numéro de la ligne et de la colonne du bloc k utilisé pour construire le CL numéro q .

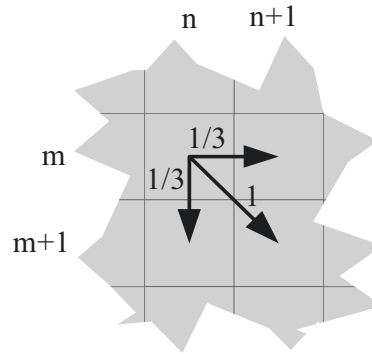


Figure 3.13 Déplacements possibles pour un chemin local (CL)

$$\alpha_{k,q}(m, n) = \begin{cases} 1/3, & \text{pour les déplacements horizontaux et verticaux} \\ 1, & \text{pour les déplacements diagonaux} \end{cases} \quad (3.6)$$

Une force $\gamma_{k,q}$ est trouvée pour chacun des Q CL possibles en considérant les éléments de la matrice de similarités $P_{k,q}$ et le coefficient $\alpha_{k,q}$ associée au déplacement utilisé pour ce CL selon l'équation 3.7.

$$\gamma_{k,q} = \sum_{\substack{m,n \in \text{chemin } q, \\ \text{bloc } k}} P_{k,q}(m, n) \alpha_{k,q}(m, n) \quad (3.7)$$

Finalement, le CLPO du bloc k correspond simplement au CL ayant la plus grande force $\gamma_{k,q}$ à l'intérieur du bloc k selon l'équation 3.8.

$$CLPO = \arg \max_q (\gamma_{k,q}) \quad (3.8)$$

Afin d'aider à la compréhension, la figure 3.14 présente tous les CL possibles pour un bloc de dimension 3×3 . Considérant l'exemple de la figure 3.12, le CLPO correspondrait logiquement au CL #6 de la figure 3.14, puisque la valeur de pertinence est proportionnelle à l'intensité du pixel.

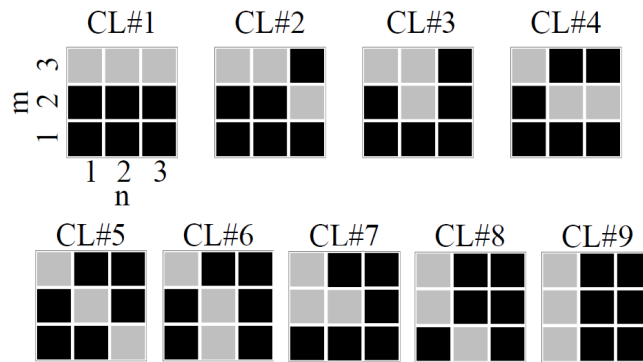


Figure 3.14 Tous les CL possibles pour un bloc de dimension 3×3

Idéalement, pour obtenir un bon alignement, il est préférable de prendre des dimensions de bloc plus élevées que celles de la figure 3.14, tel qu'un bloc de dimension 6×6 . Il est à noter que l'utilisation de ce bloc implique un délai constant sur le temps de calcul de l'alignement puisqu'il est nécessaire de cumuler un certain nombre de notes jouées avant de pouvoir chercher le CLPO. Par exemple, pour un bloc de dimension 4×4 , il est nécessaire d'avoir cumulé 4 notes jouées avant de pouvoir trouver le CLPO.

3. Enregistrement et réservation temporaire des éléments du CLPO

La construction d'un nouveau chemin survivant (CS) est démarrée si la force $\gamma_{k,q}$ du CLPO est supérieure à un seuil fixé. Lorsque la construction d'un CS est démarrée, les coordonnées des deux premiers éléments du CLPO correspondent aux deux premiers éléments du nouveau CS. Les coordonnées des éléments 3 à M du CLPO, quant à eux, sont réservées pour ce CS, et ce, jusqu'à la fin du traitement des positions Ω_x des notes jouées j et $j + 1$. Plus précisément, pour tous les CLPO dont la première ligne du bloc k correspond à la ligne j ou $j + 1$, la valeur de pertinence $P_{k,q}(m, n)$ des éléments réservés sera fixée systématiquement à 0 pour le calcul de toutes les

forces $\gamma_{k,q}$ durant la période de la réservation. Cette réservation permet d'assurer la non-convergence de CS. Elle évite, lors de la construction des CS, que des CS passent par un même point de la matrice de similarité et que tous leurs éléments qui suivent soient, en tout point, les mêmes que les autres CS qui passent aussi par ce point.

4. Construction des chemins survivants (CS) actifs

Une fois tous les CLPO des positions Ω_x trouvés pour une note jouée j , la poursuite de la construction des chemins survivants actifs créés lors du traitement des notes jouées précédemment est entamée. Un CS peut être actif ou inactif. Un CS actif contient un nombre d'éléments indéfini, c'est-à-dire qu'il est possible de concaténer continuellement des éléments à la fin de celui-ci, et ce, tant qu'il est actif. Un CS inactif, quant à lui, est simplement un CS pour lequel la construction a été arrêtée définitivement à un certain moment lors de la construction.

D'abord, pour chacun des chemins survivants actifs créés avant la note jouée courante j , le CLPO est recherché en utilisant le dernier élément de ce CS comme position 1,1 du bloc d'analyse k . La force $\gamma_{k,q}$ du CLPO associé au CS est comparée à un seuil. Si le seuil n'est pas dépassé, le CS devient inactif. Dans le cas où le seuil est dépassé, les coordonnées du deuxième élément du CLPO sont simplement concaténées au CS courant pour devenir le dernier élément de ce CS et les éléments 3 à M du CLPO sont réservés pour ce CS tel que discuté à l'étape précédente.

Il est à noter que lorsqu'un musicien désire sauter d'un endroit à l'autre de la partition, il peut être intéressant de rendre tous les CS inactifs lorsqu'un silence de plusieurs secondes se produit.

La figure 3.15 présente un exemple de matrice de CS où les CS ont été successivement créés selon l'ordre de leur numérotation. Dans cet exemple, seul le CS no. 6 est actif, car la construction des autres CS a été stoppée et la construction du CS no. 3 a débuté à la note jouée 4.

5. Retrait des CS inactifs ne respectant pas les spécifications minimales

Cette étape consiste à retirer tous les CS inactifs contenant un nombre d'éléments inférieur à un nombre minimal requis et à retirer les CS inactifs dont la moyenne des valeurs de similarités P_{ij} est inférieure à un seuil fixé. Ainsi, seule une faible partie des CS est conservée pour l'alignement.

La figure 3.16 présente un exemple où les CS de la figure 3.15 ne respectant pas les spécifications minimales ont été retirés. Pour cet exemple, le nombre d'éléments minimal est ajusté à 4 et la moyenne des valeurs de similarités minimale est ajustée à 0.3, et ce, selon des valeurs arbitraires sélectionnées pour fin d'illustration seulement. Ainsi, les CS 3 et 7 ont été retirés, car ils possédaient une longueur inférieure à la

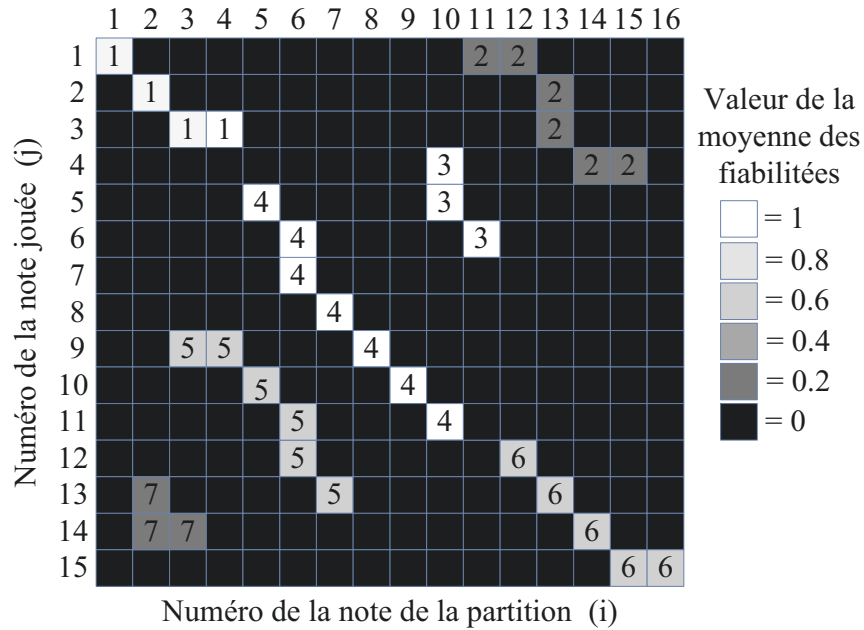


Figure 3.15 Exemple de matrice de chemins survivants

longueur minimale de 4. De plus, le CS 2 a aussi été retiré, car sa moyenne des valeurs de pertinence est inférieure au seuil fixé.

6. Recherche des meilleurs chemins survivants

Une fois toutes les étapes précédentes terminées pour une note jouée j , le meilleur alignement pour les notes jouées 1 à j est trouvé. Pour ce faire, tous les CS où les notes jouées j ne se chevauchent pas sont conservés en commençant la sélection selon l'ordre décroissant de longueur des CS où la longueur correspond au nombre de notes jouées contenues dans le CS. Lorsque les CS sont de même longueur, ils sont choisis selon l'ordre décroissant de la moyenne des valeurs de similarités contenues dans le chemin survivant. La liste de tous les CS conservés correspond à l'alignement.

Par contre, lorsque le premier CS utilisé pour construire l'alignement est un mauvais choix, l'alignement résultant est généralement très mauvais. Par conséquent, pour éviter ce problème, l'étape 6 peut être répétée un certain nombre de fois, mais en partant par des chemins survivants différents, et ce, en conservant l'ordre utilisé pour la sélection. Par exemple, si les deux CS les plus longs sont d'une longueur de 5 et de 4, l'étape 6 utilisera le CS de longueur 5 en premier pour créer un premier alignement. Ensuite, l'étape 6 sera répétée, mais en commençant cette fois par le CS d'une longueur 4 pour créer le deuxième alignement. À la fin, l'alignement qui aura aligné le plus de notes jouées sera l'alignement de sortie du système. Finalement, les étapes 1 à 6 sont répétées jusqu'à ce que toutes les notes jouées aient été traitées.

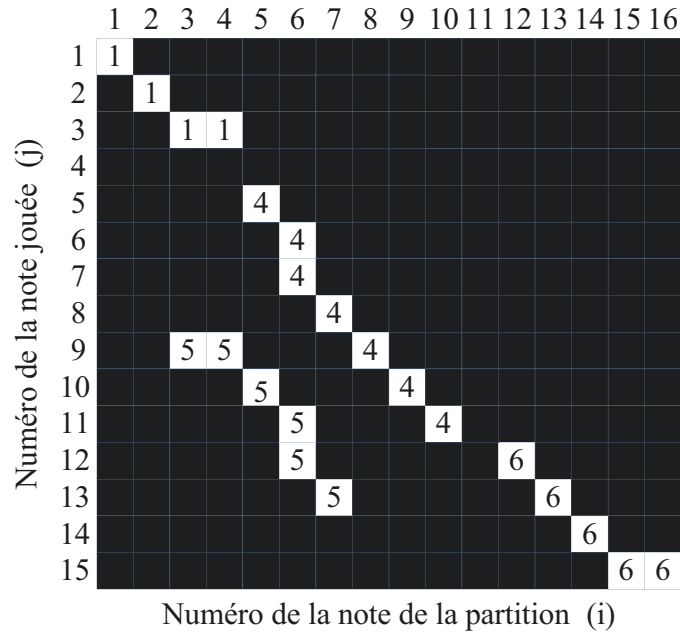


Figure 3.16 CS après le retrait des CS ne respectant pas les spécifications minimales

Pour améliorer les résultats, il est possible de permettre un petit chevauchement lors de la recherche des meilleurs CS. Par contre, ce chevauchement doit être très petit pour éviter de choisir des mauvais CS.

Selon l'exemple de la figure 3.16, le plus long CS (no. 4) est d'abord conservé. Ensuite, le deuxième plus long CS (no. 5) n'est pas conservé, car il se chevauche avec les notes jouées 9 à 11 du CS conservé no. 4. Finalement, les CS 6 et 1 sont conservés, car ils ne se chevauchent pas avec le CS conservé no. 4. L'alignement résultant de l'exemple de la figure 3.15 est présenté à la figure 3.17.

Pour terminer, il est possible d'améliorer les performances de l'algorithme EDTW en démarrant la recherche des CLPO des positions Ω_x de l'étape 2 en suivant l'ordre décroissant des valeurs de similarités P_{ij} associées à chacune des positions Ω_x tout en continuant de réserver temporairement les positions 3 à M pour ces CLPO. Une fois tous les CLPO trouvés pour une note jouée j , toutes les réservations associées à ces CLPO sont effacées. Ensuite, les CLPO de ces mêmes positions Ω_x sont recherchées à nouveau, cette fois selon l'ordre décroissant des forces $\gamma_{k,l}$ obtenues à l'étape précédente, et ce, toujours en réservant temporairement les positions 3 à M pour les nouveaux CS résultants. Ces changements permettent d'améliorer la précision du démarrage des nouveaux CS en évitant d'utiliser les mauvais éléments de la matrice pour démarrer les nouveaux CS.

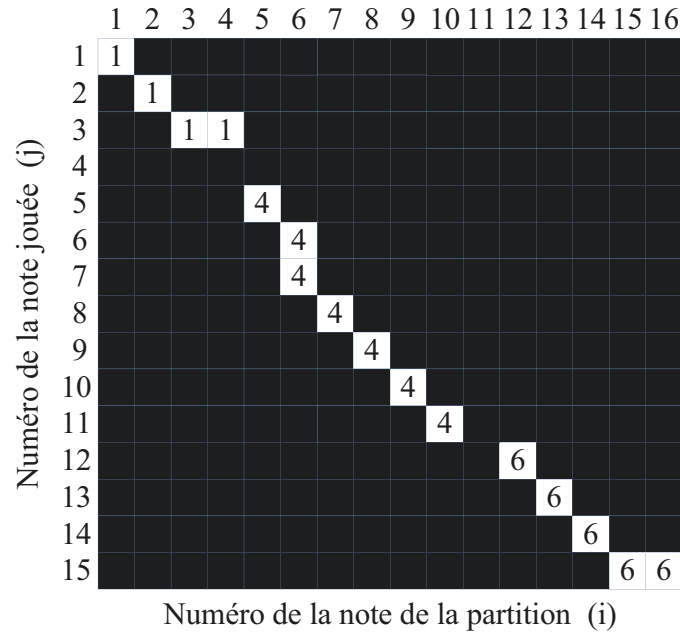


Figure 3.17 CS sélectionné pour l'alignement

Comme il y a moins d'éléments dans la matrice de similarité, la variation proposée de l'algorithme DTW, nommée EDTW, permet de trouver l'alignement optimal en faisant une recherche étendue à travers la matrice de similarités. Cette recherche étendue donne la liberté au musicien de commencer à jouer où il le désire dans la partition de musique, contrairement aux systèmes basés sur l'algorithme DTW où la première et la dernière note jouée par le musicien doivent correspondre à la première et dernière note de la partition respectivement. De plus, cette variation donne aussi la liberté au musicien de sauter d'un endroit à l'autre à travers la partition de musique du moment que les segments joués soient suffisamment long (typiquement, quelques notes suffisent). Finalement, cette adaptabilité à pouvoir trouver où le musicien joue donne la possibilité de trouver un bon alignement même à la suite d'un ou plusieurs segments incorrectement alignés. Ainsi, l'algorithme EDTW offre une grande résistance aux erreurs de performances faites par le musicien et aux erreurs de traitement du signal. Pour terminer, comme le musicien entend des notes et non pas des trames, l'algorithme EDTW proposé se rapproche davantage du comportement d'un musicien lorsque celui-ci suit une partition de musique.

CHAPITRE 4

APPLICATIONS EXISTANTES

Il va sans dire que certains algorithmes de traitement du signal et d'alignement ne seront probablement jamais accessibles au public, et ce, afin de conserver propriété intellectuelle parfois nécessaire au développement des entreprises. Toutefois, il est jugé pertinent d'analyser des applications existantes nécessitant divers algorithmes de ce type. D'ailleurs, certaines de ces applications se rapprochent de l'objet visé par ce projet de recherche.

Deux groupes de recherche ont tenté de réaliser un système d'apprentissage de la musique où le musicien doit jouer une partition de musique affichée à l'écran et dont le système affiche sur la partition des symboles pour indiquer les endroits où le musicien a fait des erreurs de performance. Les systèmes réalisés par ces deux groupes de recherche se nomment *Vemus* [20] et *interactive digital violin tutor* (iDVT) [42]. Il est à noter que ces deux systèmes n'invalident pas ce projet de recherche puisqu'ils ont été conçus pour des pièces de musique de niveau débutant comparativement à ce projet de recherche qui vise un fonctionnement pour des pièces de niveau avancé.

Il existe aussi un logiciel commercialisé de ce type, le logiciel *SmartMusic*. Cependant, comme il sera discuté ultérieurement, ce logiciel impose au musicien de suivre un tempo bien précis facilitant grandement le suivi automatique de la partition de musique. Un groupe d'étudiants du baccalauréat de l'université de Sherbrooke a aussi réalisé un logiciel de ce type comme projet de fin de baccalauréat, ce projet est connu sous le nom de professeur virtuel de musique (PVM). Néanmoins, ce logiciel était peu performant et incapable de traiter de pièces de niveau avancé. Pour terminer, il est jugé pertinent d'analyser l'application nommée *Tonara* puisque cette dernière permet de tourner automatiquement des pages de partitions de musique, et ce, pour des pièces polyphoniques de niveau avancé. Les sections suivantes présentent en plus de détails chacune de ces applications.

4.1 *SmartMusic*

Le logiciel *SmartMusic* de la compagnie *MakeMusic* est disponible sur le marché et permet de trouver les erreurs de performance à partir d'une performance musicale capturée par un microphone. Nous avons acheté ce logiciel afin d'évaluer sa performance. La figure 4.1 présente la fenêtre principale du logiciel.

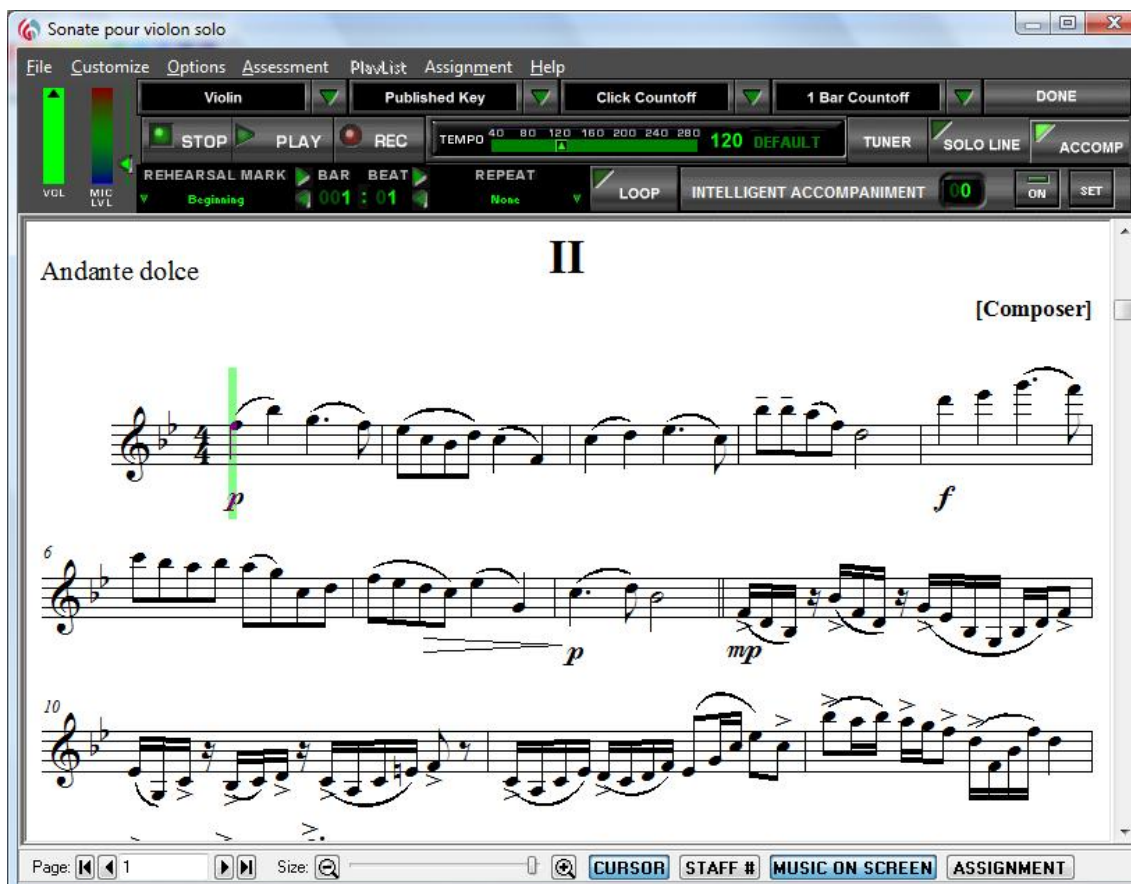
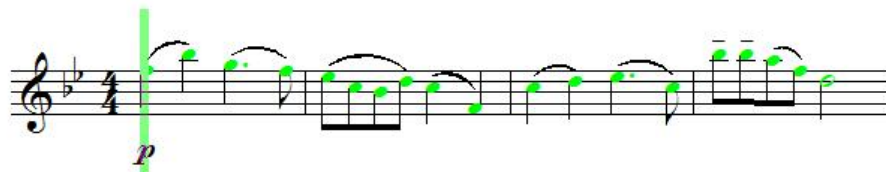
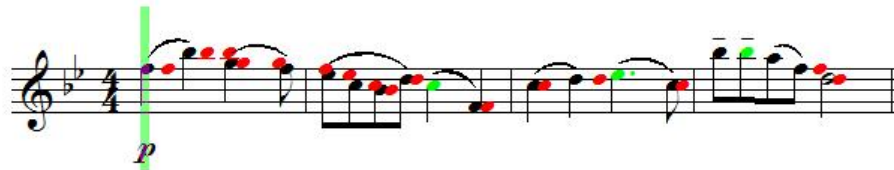


Figure 4.1 Fenêtre principale du logiciel SmartMusic de MakeMusic

Nous avons d'abord constaté que le logiciel ne suivait pas le musicien lorsque celui-ci jouait la partition de musique. En fait, c'est le contraire, c'est le musicien qui doit suivre l'application en jouant au bon moment les notes marquées par un curseur. Les figures 4.2(a) et 4.2(b) permettent d'illustrer ceci. D'abord, la figure 4.2(a) présente les erreurs trouvées par l'application lorsqu'une performance musicale préenregistrée est démarrée de façon à ce que les notes jouées arrivent au moment indiqué par le curseur. Ensuite, la figure 4.2(b) présente les erreurs trouvées par l'application lorsque la performance musicale préenregistrée de la figure 4.2(a) est démarrée trop tard de façon à ce que les notes jouées arrivent environ 300 msec après le moment indiqué par le curseur de l'application. On remarque que lorsque l'enregistrement est démarré au bon moment, le logiciel ne détecte pas d'erreur de performance. Par contre, lorsque ce même enregistrement est démarré en retard, l'application trouve de multiples erreurs de performance malgré que le segment de musique soit correctement joué.



(a) Notes jouées au moment indiqué par le curseur de SmartMusic



(b) Notes jouées 300 msec après le moment indiqué par le curseur de SmartMusic

Figure 4.2 Exemple d'erreurs de performance trouvées par le logiciel SmartMusic pour un même segment audio de musique, mais démarré à temps et trop tard (les notes vertes (en gris pâle) sont des notes jouées correctement et les notes rouge (en gris foncé) sont des notes jouées trop tôt ou trop tard.)

Idéalement, le musicien doit apprendre à maintenir le rythme par lui-même. Par analogie, lui donner le rythme c'est comme tenir un bébé qui apprend à marcher pour qu'il ne tombe pas. Un jour, il faut lâcher le bébé pour qu'il apprenne à marcher de lui-même. D'ailleurs, cette contrainte peut désenchanter beaucoup de musiciens puisqu'elle fait clairement entrave aux sentiments de liberté nécessaires pour la motivation.

L'imposition du rythme diminue considérablement la complexité du traitement du signal nécessaire pour extraire l'information musicale du signal audio numérique. En effet, comme l'application connaît déjà la note désirée, et ce, pour chaque instant, elle n'a qu'à vérifier

si cette note est présente à l’instant attendu. La figure 4.3 illustre un exemple de sortie du logiciel SmartMusic pour un segment audio de violon très rapide. Il est à noter que pour une application qui ne sait pas a priori quand les notes commencent et se terminent, l’information musicale contenue dans ce segment est très difficile à extraire. D’ailleurs, ce segment audio numérique était problématique pour le système réalisé lors de la maîtrise de l’auteur [21]. Pour le sujet de recherche, le rythme ne sera pas imposé et les algorithmes de traitement du signal devront fonctionner relativement bien en présence de signaux audio rapides tels que celui utilisé pour la figure 4.3.

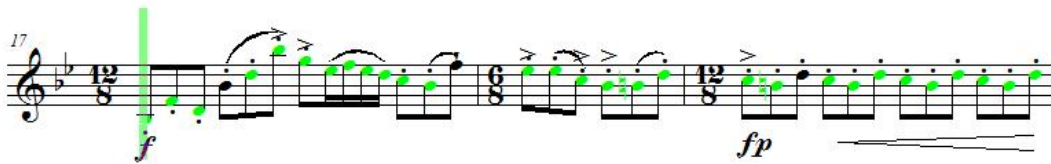


Figure 4.3 Exemple d’erreurs de performance trouvées par le logiciel SmartMusic pour un segment très rapide de violon joué à environ 120 battements à la minute (les notes en noir sont des notes non détectées)

Le logiciel SmartMusic détecte seulement quelques erreurs de performance, c’est-à-dire les notes sautées, les notes jouées incorrectement, les notes jouées trop tôt et les notes jouées trop tard. Idéalement, pour être complet, un logiciel devrait aussi de détecter d’autres erreurs de performance telles que des erreurs sur la dynamique (force du son) et le rythme. Pour terminer, si les notes sont jouées au bon moment, la performance du logiciel *SmartMusic* est acceptable pour détecter les erreurs de performance de base.

4.2 Professeur virtuel de musique

Le professeur virtuel de musique (PVM) de la figure 4.4 correspond au projet de fin de baccalauréat d’un groupe composé de 8 étudiants de l’Université de Sherbrooke. Il a été développé à la suite d’une offre de projet de la part des directeurs de l’auteur de cette thèse. L’objectif du projet était de réaliser un logiciel similaire à SmartMusic et de fournir le programme source aux chercheurs de l’université dans le domaine. Ceci afin que ces derniers puissent y interfacer leurs codes pour présenter concrètement leurs résultats à des néophytes du traitement du signal à l’aide d’une interface graphique similaire à celle retrouvée dans un produit commercial.

L’application PVM est assez complète au niveau des fonctionnalités, des différentes configurations possibles et du nombre de types d’erreurs de performance pouvant être affichées. Elle comprend plusieurs outils pratiques tels qu’un accordeur et un métronome.

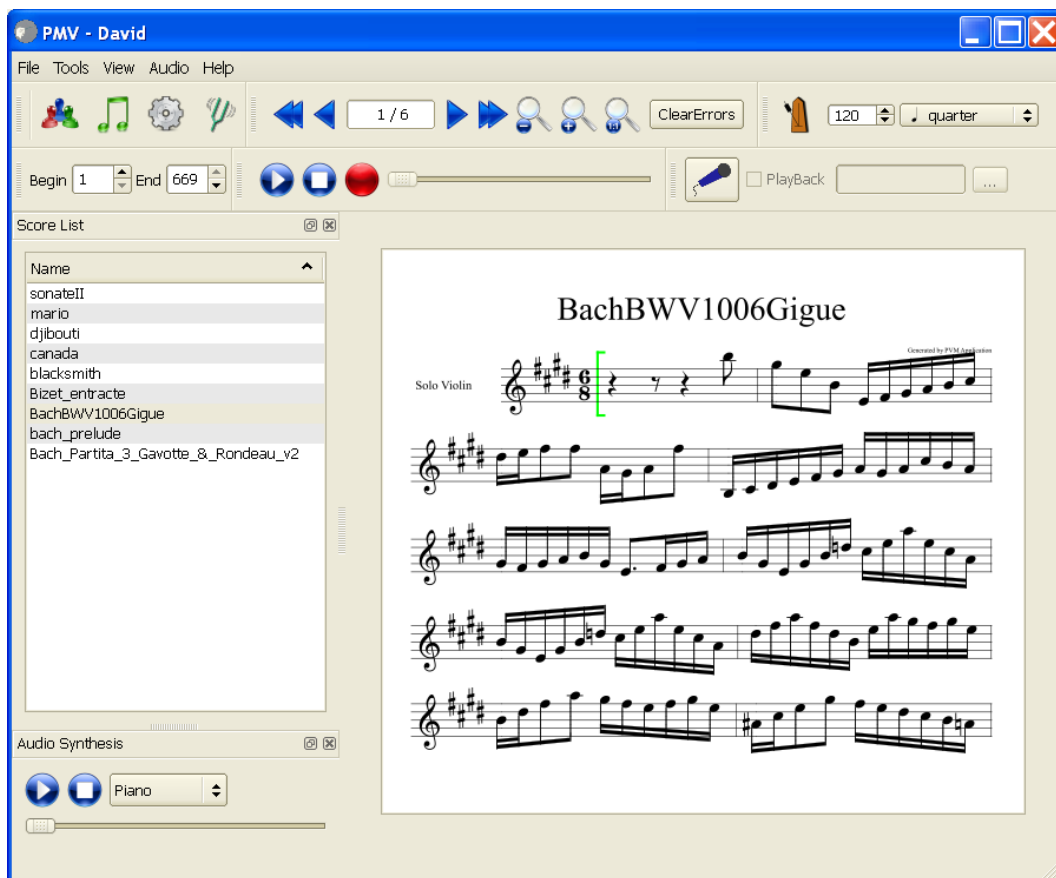


Figure 4.4 Fenêtre principale de l'application PVM

Néanmoins, elle nécessite encore un peu de développement au niveau de l’affichage de la partition et de la robustesse en général.

Les algorithmes de traitement du signal utilisés pour faire l’alignement de la partition sont essentiellement les algorithmes réalisés lors des travaux antérieurs de recherche de maîtrise [21] de l’auteur de cette thèse. Toutefois, l’équipe PVM a ajouté en parallèle un algorithme d’alignement en direct utilisant comme entrée les notes détectées par l’algorithme de transcription automatique des travaux antérieurs de recherche de maîtrise. Ceci s’explique par le fait que l’algorithme d’alignement de la maîtrise nommé EDTW (*Extended Dynamic Time Warping*) calcule l’alignement optimal avec un délai constant de quelques notes. En effet, ce délai constant ne permet pas à l’application de suivre en direct avec un curseur l’endroit joué par le musicien. Toutefois, la présence de l’algorithme EDTW dans l’application PVM donne la liberté au musicien de jouer le segments de musique qu’il désire, et ce, sans devoir préalablement informer le système des segments qu’il jouera. Cette liberté donne un avantage important par rapport à l’application SmartMusic.

Pour leur part, les algorithmes de détection d’erreurs de performance sont des algorithmes primitifs réalisés antérieurement par l’auteur de cette thèse et le groupe PVM, et ce, dans le but unique de démontrer le fonctionnement du système complet. La figure 4.5 présente un exemple de résultats obtenus en utilisant le même segment audio que celui de la figure 4.2(b) démontrant la performance du logiciel SmartMusic pour un segment de musique simple et joué par un violoniste professionnel.

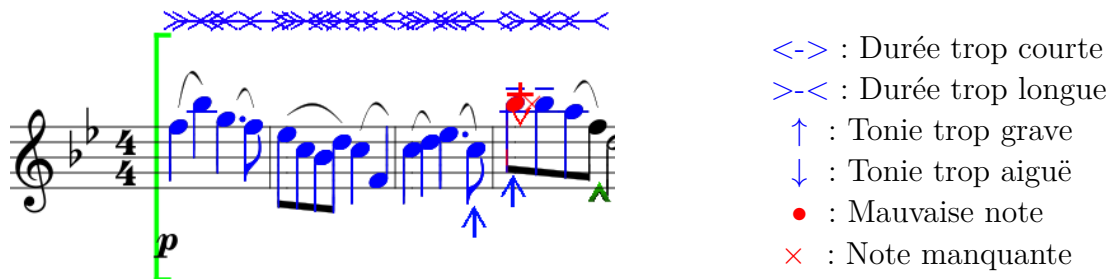


Figure 4.5 Exemple d’erreurs de performance trouvées par l’application PVM

Les notes en bleu avec des flèches horizontales au-dessus représentent des notes jouées avec une mauvaise durée. Le nombre important de ces notes s’explique par le fait que l’application PVM vérifie la précision de la durée des notes en fonction du rythme indiqué sur le métronome de l’application. Par conséquent, si le rythme n’est pas exactement celui attendu, toutes les notes vont se retrouver erronées au niveau de la durée, ce qui a été le cas pour l’exemple de la figure 4.5. À ce niveau, il serait probablement préférable que

l'application s'adapte au rythme du musicien. On remarque aussi la présence de deux flèches bleues pointant vers le haut. Ce symbole indique que les deux notes n'ont pas été jouées avec une bonne justesse et le sens indique qu'elles devaient être jouées avec une tonalité légèrement plus haute. La note en rouge indique que la note n'a pas été jouée correctement et la croix indique qu'une note n'a pas été jouée. Comme le segment joué ne contenait aucune erreur, il est évident que l'application PVM n'a pas encore atteint sa maturité. On remarque aussi que les erreurs ne sont pas affichées avec clarté. Idéalement, les erreurs affichées devraient être choisies adéquatement, et ce, dans le but de ne pas saturer et décourager le musicien.

4.3 VEMUS

Le projet de recherche VEMUS [20] est anciennement connu sous le nom de IMUTUS [67]. IMUTUS avait comme objectif de produire un logiciel d'enseignement de la flûte pour débutants similaire à DVT. Le projet a été terminé en 2008 [66] et les travaux de recherches publiés sont essentiellement au niveau des fonctionnalités de l'application. Le projet a été transféré à un autre groupe de recherche nommé VEMUS. Pour VEMUS, les publications actuelles sont aussi au niveau des fonctionnalités de l'application. Une version bêta du logiciel a été trouvée, cette version a été installée sur trois ordinateurs avec trois systèmes d'exploitation Windows différents. Malheureusement, aucune de ces installations ne s'est avérée fonctionnelle. Par conséquent, l'évaluation de cette application se base que sur les articles scientifiques publiés à son sujet.

La fenêtre principale de l'application IMUTUS est présentée à la figure 4.6. Essentiellement, le projet IMUTUS affiche des erreurs de performance sur la partition de musique, donne des conseils pour améliorer le jeu et donne un pointage sous forme d'étoiles pour la performance générale. Pour le cas de la figure 4.6, le pointage pour la performance est de 2 étoiles. Les principaux résultats fournis sont au niveau de l'utilisation. Les tests ont été faits avec 12 étudiants dont 6 utilisaient le logiciel IMUTUS. Après trois semaines de pratique avec IMUTUS, les étudiants ont répondu à un questionnaire. En conclusion, les étudiants ont trouvé l'application facile d'utilisation et amusante. Le groupe ayant utilisé IMUTUS a fait beaucoup moins d'erreurs de performance que le groupe de contrôle [67].

Le projet de recherche VEMUS est financé par la commission européenne et possède 8 partenaires provenant de 6 pays différents [20]. Tout comme pour IMUTUS, les erreurs de performances sont affichées sur la partition de musique. Par contre, les annotations sont essentiellement ajoutées par un professeur de musique distant. Pour l'instant, la partition sert principalement de moyen de communication entre un groupe collaboratif d'étudiants



Figure 4.6 Fenêtre principale du logiciel IMUTUS [67]

et un professeur de musique. Les annotations pouvant être ajoutées par le professeur sont de type textuel, binettes (émoticons), manuelles, sonores et graphiques tels que l’affichage de la fréquence fondamentale et l’enveloppe temporelle. Le projet semble ambitieux et prometteur, mais aucun résultat n’a été émis jusqu’ici et le traitement du signal ne semble pas être un point d’intérêt pour l’instant.

4.4 Digital Violin Tutor (DVT)

Le projet *Digital Violin Tutor* (DVT) avait pour objectif de créer un logiciel d’apprentissage pour violonistes débutants. La figure 4.7 présente la fenêtre principale de l’application DVT [85]. Les erreurs de performances sont affichées à l’aide du graphique 2D situé à gauche et au bas de la fenêtre. Les notes correspondent à des rectangles positionnés dans le temps vis-à-vis d’une touche de piano correspondant à la tonie de la note. L’application affiche aussi la position des notes sur la touche d’un violon, présente une vidéo du professeur jouant la prescription demandée et synthétise une vidéo virtuelle d’un musicien en fonction des notes à exécuter et selon le doigté attendu, et ce, avec un beau paysage en arrière-plan dans le but d’aider le musicien à relaxer.

Contrairement à SmartMusic, le musicien n’est pas contraint à suivre un rythme imposé. Les figures 4.8(a) et 4.8(b) provenant de [85] illustrent un exemple de résultat avant et après l’alignement où les notes en vert (en gris) correspondent à la prescription, les notes en bleu (en gris foncé) aux notes jouées incorrectement et les notes en jaune (en gris pâle) aux notes jouées correctement. En observant les résultats après l’alignement de la figure 4.8(b), l’interprétation est la suivante : le musicien a omis de jouer la première note

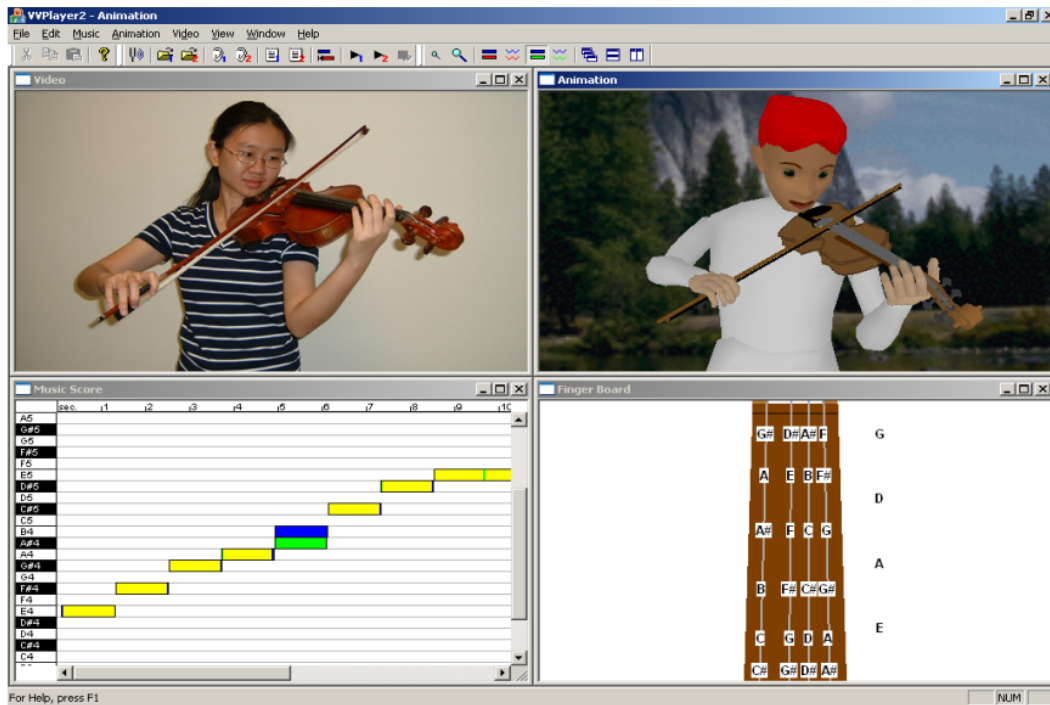
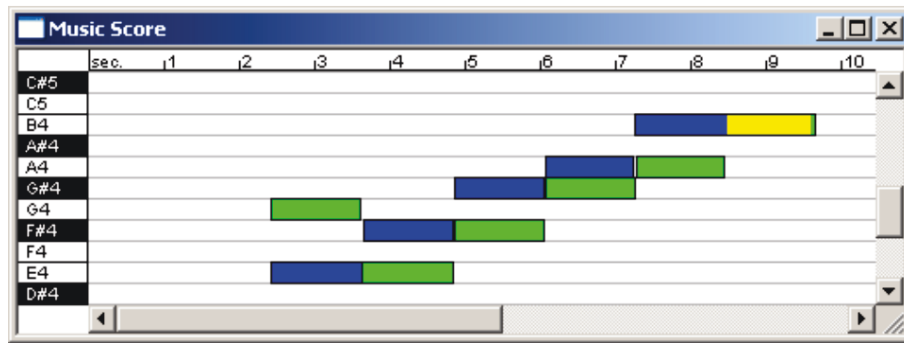


Figure 4.7 Fenêtre principale de l'application DVT [85]

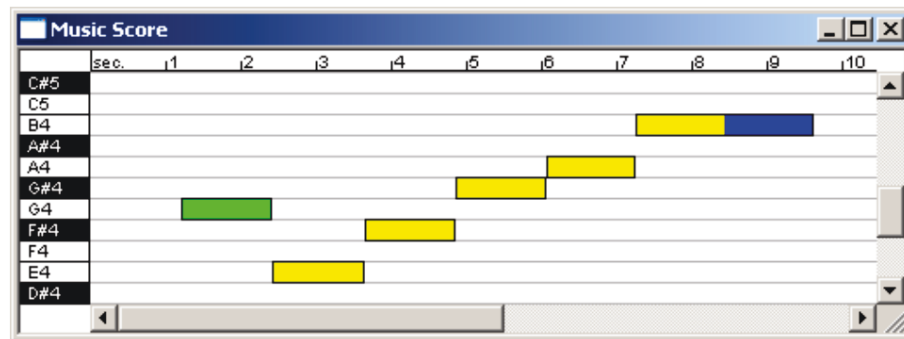
prescrite et a joué trop longtemps la dernière note prescrite. Par contre, en observant les résultats avant l'alignement de la figure 4.8(a), l'interprétation est la suivante : le musicien a joué incorrectement les 5 premières notes prescrites et la dernière note a été jouée correctement. Ces résultats démontrent clairement l'importance de l'alignement, puisque les résultats obtenus après l'alignement sont beaucoup plus représentatifs de la performance de musicien.

Comme l'application DVT n'est pas distribuée, il n'est pas possible d'évaluer directement sa performance. Par conséquent, l'AETM utilisé par DVT et décrit par l'article [41] a été codé pour aider à l'évaluation. Le fonctionnement de l'AETM a été expliqué à la sous-section 2.1. La figure 4.9 illustre un exemple de résultats obtenus à partir de l'implémentation de l'article [41] en utilisant comme entrée le même signal audio que celui utilisé pour la figure 4.3. Il est à noter que les lignes horizontales de la figure 4.9 correspondent aux fréquences fondamentales nominales des notes qui devraient être trouvées par l'AETM s'il était parfait et la ligne continue correspond à la fréquence fondamentale estimée par l'AETM.

Les résultats de la figure 4.9 démontrent que l'AETM utilisé par l'application DVT ne détecte pas toutes les notes contenues dans le segment audio et détectent quelques notes en trop. Bien que les résultats semblent inférieurs à ceux de SmartMusic, ceci n'implique



(a) Avant l'alignement



(b) Après l'alignement

Figure 4.8 Exemple d'alignement obtenu par DVT [85]

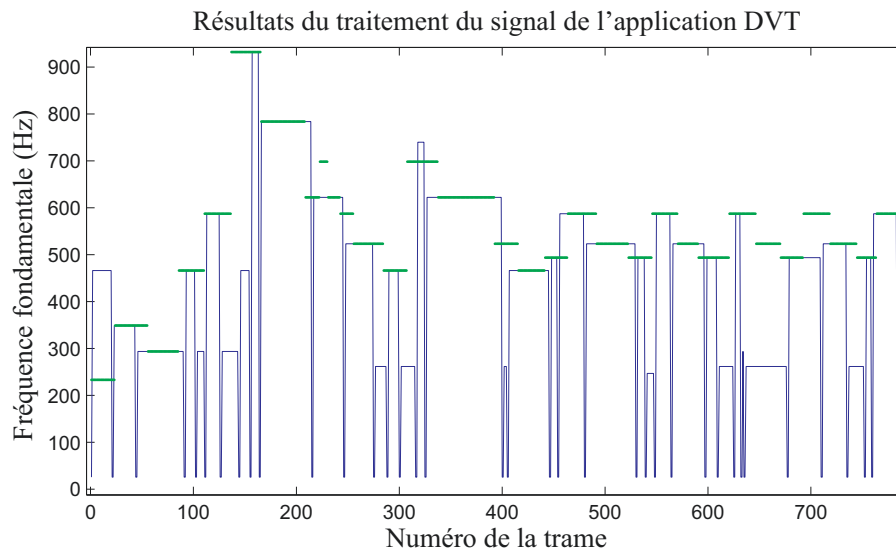


Figure 4.9 Exemple de résultats obtenus par l'AETM utilisé par l'application DVT. Les lignes horizontales en vert (en gris pâle) correspondent aux fréquences fondamentales que l'algorithme aurait dû détecter et la courbe en bleu (en gris) correspond à la fréquence fondamentale trouvée par l'AETM

pas que l'AETM est inférieur, puisque l'AETM de DVT ne connaît pas à priori les notes attendues. Par conséquent, l'AETM peut se tromper en estimant les notes contenues dans le segment.

Le sujet de recherche DVT porte maintenant le nom de iDVT (*interactive Digital Violin Tutor*, en anglais). L'application iDVT est similaire à l'application DVT, mais avec du traitement vidéo ajouté [42]. Pour ce faire, une caméra capte le doigté du violoniste et une autre caméra capte le mouvement de l'archet. La capture vidéo a pour objectif d'améliorer la performance de la segmentation des notes, c'est-à-dire qu'elle permet de détecter plus efficacement le début et la fin des notes jouées par le musicien. Les résultats de iDVT démontrent que la fusion de l'audio avec la vidéo permet d'améliorer la détection des bons débuts de notes de 10 % et de diminuer de 8 % la détection des mauvais débuts de notes [82]. Ces résultats sont très significatifs et démontrent l'intérêt de converger les médias tels que la vidéo et l'audio. Par contre, pour utiliser iDVT, le musicien devra acheter deux caméras vidéo ainsi que deux trépieds pour ajuster les caméras. De plus, il devra prendre un certain temps pour configurer et ajuster correctement les caméras et la luminosité.

4.5 Tonara

Actuellement, l'application Tonara est une application dédiée pour les produits Ipad de la compagnie Apple. Cette application permet essentiellement de visualiser des partitions de musique sur une tablette. Il est jugé pertinent de présenter cette application puisqu'elle est dotée d'une méthode d'alignement relativement performante. La figure 4.10 présente une capture d'écran de l'application Tonara où il est possible d'observer un petit curseur à la troisième mesure¹ de la partition de musique. Ce curseur indique la position estimée par la méthode d'alignement intégrée dans l'application correspondant à la position de la note jouée par le musicien.

Selon les spécifications du produit, l'application fonctionne pour des pièces de niveau débutant jusqu'à professionnel. La méthode d'alignement utilisée peut supporter des variations de tempo. Par ailleurs, le musicien peut arrêter de jouer un moment et ensuite continuer à jouer sans que le suivi de la partition soit perturbé.

Suite à une évaluation de l'application réalisée à partir de tests, et ce, en jouant directement du violon et en utilisant des enregistrements audios professionnels de violon, les constatations suivantes ont été relevées :

1. Une mesure correspond à un segment de notes d'une partition de musique borné par deux barres verticales pour lequel la signature temporelle est spécifiée.



The image is a screenshot of a mobile application interface for music. At the top, there is a navigation bar with the text "Mes partitions" on the left and "Tonara Magic" on the right, accompanied by a blue toggle switch. Below the navigation bar, the title of the piece is displayed: "Solo Violin Sonata no. 2 in A minor, BWV 1003 - 3rd mvt." The tempo is marked "Andante" and the composer is "J. S. Bach (1685-1750)". The main area of the screen shows six staves of musical notation in treble clef, 3/4 time signature. The notation includes various rhythmic values, accidentals, and dynamic markings. The first staff starts with a treble clef and a 3/4 time signature. The second staff begins with a measure rest and a fermata. The third staff has a measure rest. The fourth staff has a measure rest. The fifth staff has a measure rest. The sixth staff has a measure rest. The seventh staff has a measure rest. The eighth staff has a measure rest. The ninth staff has a measure rest. The tenth staff has a measure rest. The eleventh staff has a measure rest. The twelfth staff has a measure rest. The thirteenth staff has a measure rest. The fourteenth staff has a measure rest. The fifteenth staff has a measure rest. The sixteenth staff has a measure rest. The seventeenth staff has a measure rest. The eighteenth staff has a measure rest. The nineteenth staff has a measure rest. The twentieth staff has a measure rest. The twenty-first staff has a measure rest. The twenty-second staff has a measure rest. The twenty-third staff has a measure rest. The twenty-fourth staff has a measure rest. The twenty-fifth staff has a measure rest. The twenty-sixth staff has a measure rest. The twenty-seventh staff has a measure rest. The twenty-eighth staff has a measure rest. The twenty-ninth staff has a measure rest. The thirtieth staff has a measure rest. The thirty-first staff has a measure rest. The thirty-second staff has a measure rest. The thirty-third staff has a measure rest. The thirty-fourth staff has a measure rest. The thirty-fifth staff has a measure rest. The thirty-sixth staff has a measure rest. The thirty-seventh staff has a measure rest. The thirty-eighth staff has a measure rest. The thirty-ninth staff has a measure rest. The fortieth staff has a measure rest. The forty-first staff has a measure rest. The forty-second staff has a measure rest. The forty-third staff has a measure rest. The forty-fourth staff has a measure rest. The forty-fifth staff has a measure rest. The forty-sixth staff has a measure rest. The forty-seventh staff has a measure rest. The forty-eighth staff has a measure rest. The forty-ninth staff has a measure rest. The fiftieth staff has a measure rest. The fifty-first staff has a measure rest. The fifty-second staff has a measure rest. The fifty-third staff has a measure rest. The fifty-fourth staff has a measure rest. The fifty-fifth staff has a measure rest. The fifty-sixth staff has a measure rest. The fifty-seventh staff has a measure rest. The fifty-eighth staff has a measure rest. The fifty-ninth staff has a measure rest. The sixtieth staff has a measure rest. The sixty-first staff has a measure rest. The sixty-second staff has a measure rest. The sixty-third staff has a measure rest. The sixty-fourth staff has a measure rest. The sixty-fifth staff has a measure rest. The sixty-sixth staff has a measure rest. The sixty-seventh staff has a measure rest. The sixty-eighth staff has a measure rest. The sixty-ninth staff has a measure rest. The seventieth staff has a measure rest. The seventy-first staff has a measure rest. The seventy-second staff has a measure rest. The seventy-third staff has a measure rest. The seventy-fourth staff has a measure rest. The seventy-fifth staff has a measure rest. The seventy-sixth staff has a measure rest. The seventy-seventh staff has a measure rest. The seventy-eighth staff has a measure rest. The seventy-ninth staff has a measure rest. The eightieth staff has a measure rest. The eighty-first staff has a measure rest. The eighty-second staff has a measure rest. The eighty-third staff has a measure rest. The eighty-fourth staff has a measure rest. The eighty-fifth staff has a measure rest. The eighty-sixth staff has a measure rest. The eighty-seventh staff has a measure rest. The eighty-eighth staff has a measure rest. The eighty-ninth staff has a measure rest. The ninetieth staff has a measure rest. The hundredth staff has a measure rest. The hundred and first staff has a measure rest. The hundred and second staff has a measure rest. The hundred and third staff has a measure rest. The hundred and fourth staff has a measure rest. The hundred and fifth staff has a measure rest. The hundred and sixth staff has a measure rest. The hundred and seventh staff has a measure rest. The hundred and eighth staff has a measure rest. The hundred and ninth staff has a measure rest. The hundred and tenth staff has a measure rest. The hundred and eleventh staff has a measure rest. The hundred and twelfth staff has a measure rest. The hundred and thirteenth staff has a measure rest. The hundred and fourteenth staff has a measure rest. The hundred and fifteenth staff has a measure rest. The hundred and sixteenth staff has a measure rest. The hundred and seventeenth staff has a measure rest. The hundred and eighteenth staff has a measure rest. The hundred and nineteenth staff has a measure rest. The hundred and twentieth staff has a measure rest. The hundred and twenty-first staff has a measure rest. The hundred and twenty-second staff has a measure rest. The hundred and twenty-third staff has a measure rest. The hundred and twenty-fourth staff has a measure rest. The hundred and twenty-fifth staff has a measure rest. The hundred and twenty-sixth staff has a measure rest. The hundred and twenty-seventh staff has a measure rest. The hundred and twenty-eighth staff has a measure rest. The hundred and twenty-ninth staff has a measure rest. The hundred and thirtieth staff has a measure rest. The hundred and thirty-first staff has a measure rest. The hundred and thirty-second staff has a measure rest. The hundred and thirty-third staff has a measure rest. The hundred and thirty-fourth staff has a measure rest. The hundred and thirty-fifth staff has a measure rest. The hundred and thirty-sixth staff has a measure rest. The hundred and thirty-seventh staff has a measure rest. The hundred and thirty-eighth staff has a measure rest. The hundred and thirty-ninth staff has a measure rest. The hundred and fortieth staff has a measure rest. The hundred and forty-first staff has a measure rest. The hundred and forty-second staff has a measure rest. The hundred and forty-third staff has a measure rest. The hundred and forty-fourth staff has a measure rest. The hundred and forty-fifth staff has a measure rest. The hundred and forty-sixth staff has a measure rest. The hundred and forty-seventh staff has a measure rest. The hundred and forty-eighth staff has a measure rest. The hundred and forty-ninth staff has a measure rest. The hundred and fiftieth staff has a measure rest. The hundred and fifty-first staff has a measure rest. The hundred and fifty-second staff has a measure rest. The hundred and fifty-third staff has a measure rest. The hundred and fifty-fourth staff has a measure rest. The hundred and fifty-fifth staff has a measure rest. The hundred and fifty-sixth staff has a measure rest. The hundred and fifty-seventh staff has a measure rest. The hundred and fifty-eighth staff has a measure rest. The hundred and fifty-ninth staff has a measure rest. The hundred and sixtieth staff has a measure rest. The hundred and sixty-first staff has a measure rest. The hundred and sixty-second staff has a measure rest. The hundred and sixty-third staff has a measure rest. The hundred and sixty-fourth staff has a measure rest. The hundred and sixty-fifth staff has a measure rest. The hundred and sixty-sixth staff has a measure rest. The hundred and sixty-seventh staff has a measure rest. The hundred and sixty-eighth staff has a measure rest. The hundred and sixty-ninth staff has a measure rest. The hundred and seventieth staff has a measure rest. The hundred and seventy-first staff has a measure rest. The hundred and seventy-second staff has a measure rest. The hundred and seventy-third staff has a measure rest. The hundred and seventy-fourth staff has a measure rest. The hundred and seventy-fifth staff has a measure rest. The hundred and seventy-sixth staff has a measure rest. The hundred and seventy-seventh staff has a measure rest. The hundred and seventy-eighth staff has a measure rest. The hundred and seventy-ninth staff has a measure rest. The hundred and eightieth staff has a measure rest. The hundred and eighty-first staff has a measure rest. The hundred and eighty-second staff has a measure rest. The hundred and eighty-third staff has a measure rest. The hundred and eighty-fourth staff has a measure rest. The hundred and eighty-fifth staff has a measure rest. The hundred and eighty-sixth staff has a measure rest. The hundred and eighty-seventh staff has a measure rest. The hundred and eighty-eighth staff has a measure rest. The hundred and eighty-ninth staff has a measure rest. The hundred and ninetieth staff has a measure rest. The hundred and ninety-first staff has a measure rest. The hundred and ninety-second staff has a measure rest. The hundred and ninety-third staff has a measure rest. The hundred and ninety-fourth staff has a measure rest. The hundred and ninety-fifth staff has a measure rest. The hundred and ninety-sixth staff has a measure rest. The hundred and ninety-seventh staff has a measure rest. The hundred and ninety-eighth staff has a measure rest. The hundred and ninety-ninth staff has a measure rest. The hundredth staff has a measure rest.

Figure 4.10 Capture d'écran de l'application Tonara

1. Le suivi fonctionne très bien lorsque la pièce est jouée sans fautes.
2. Le suivi peut facilement être rompu en présence de quelques erreurs musicales consécutives.
3. Le suivi tolère la présence de légères variations au niveau du tempo.
4. Le tempo ne doit pas changer trop brusquement, autrement, le curseur continue d'avancer au point de rompre complètement le suivi.
5. La méthode d'alignement automatique n'est fonctionnelle que pour les pièces disponibles dans leur librairie payante.

Bien que le suivi soit relativement performant dans les conditions idéales, il suffit que de quelques erreurs pour qu'il soit rompu de façon permanente, c'est-à-dire que le seul moyen de rétablir le suivi est de reprendre du début. De plus, il n'est pas possible de savoir si, pour une pièce de musique donnée, la méthode d'alignement utilisée nécessite un entraînement ou des ajustements préalables. Néanmoins, il est clair que pour utiliser la fonctionnalité d'alignement automatique de l'application, un musicien doit choisir une pièce parmi les pièces disponibles de la librairie Tonara, et ce, comparativement à la méthode d'alignement qui sera présentée dans cette thèse qui, pour sa part, ne nécessite ni entraînement et ajustements préalables.

Pour terminer, comparativement à celle de l'application Tonara, la méthode d'alignement présentée dans cette thèse a été conçue pour se resynchroniser en quelques secondes lorsque le suivi est rompu. Cette caractéristique peut s'avérer être un atout important puisque lorsque le suivi est rompu, le musicien n'a plus besoin d'arrêter de jouer pour indiquer à l'application où il est rendu. En effet, en présence de rupture de suivi, la méthode proposée a la capacité de chercher rapidement à travers toute la partition de musique l'endroit correspondant à ce qui est joué par le musicien. Le prochain chapitre présente en détail le fonctionnement complet du système proposé par ce projet de recherche dont l'un des modules est basé sur cette méthode d'alignement innovatrice.

CHAPITRE 5

SYSTÈME PROPOSÉ

Le système proposé repose sur l'architecture logicielle générique illustrée à la figure 5.1. Cette architecture est composée de plusieurs modules ayant différentes responsabilités.

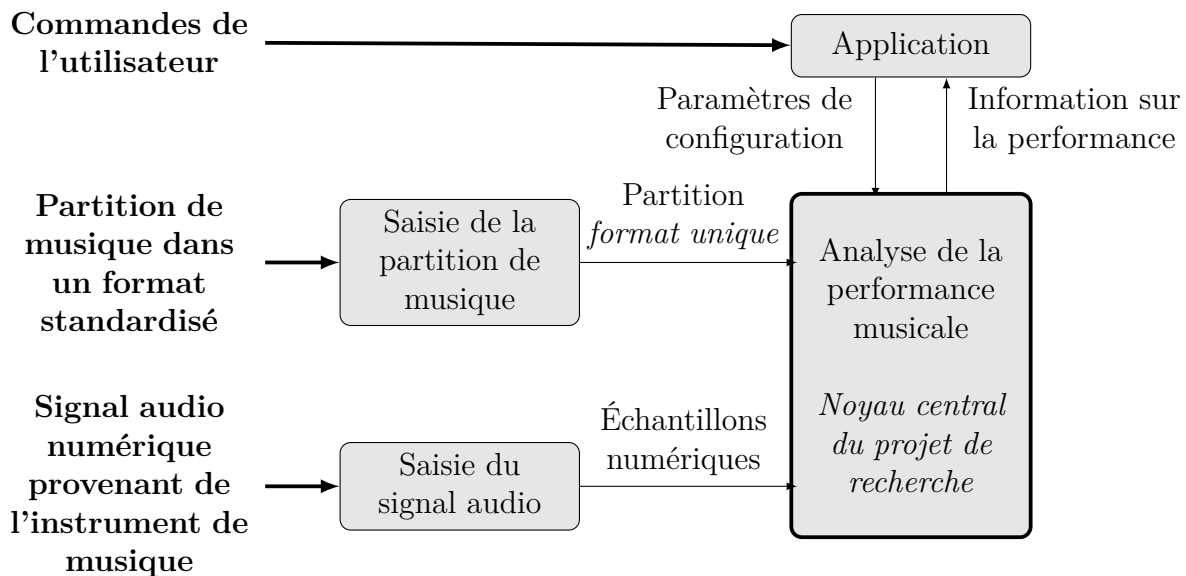


Figure 5.1 Modules de système

Le travail réalisé lors de ce projet de recherche concerne essentiellement le module *Analyse de la performance musicale*. Bien que les modules *Saisie du signal audio*, *Saisie de la partition de musique* et *Application* aient aussi été réalisés, ces derniers s'avèrent plutôt du domaine de développement technologique. De ce fait, ces trois modules sont sommairement présentés à titre informatif et ne seront pas discutés davantage en dehors de ce chapitre.

5.1 Module *Saisie du signal audio*

Le module *Saisie du signal audio* a pour fonction de contrôler le système d'acquisition du son pour transformer le signal analogique capturé par le microphone en un signal numérique composé d'échantillons audio. Il est à noter que la modulation par impulsion et codage (*Pulse-Code Modulation (PCM)*, en anglais) a été utilisée pour l'ensemble des essais réalisés tout au long de ce projet de recherche, et ce, avec une résolution de 16 bits et une fréquence d'échantillonnage de 44.1KHz. Ceci dit, aucun essai n'a été réalisé pour valider

quelle était la fréquence d'échantillonnage minimale pour assurer le bon fonctionnement du système.

5.2 Module *Saisie de la partition de musique*

Le module *Saisie de la partition de musique* a pour fonction de convertir un format de partition de musique quelconque en un format unique reconnu par le module *Analyse de la performance musicale* et conçu spécifiquement pour ce dernier. Parmi les formats existants, citons le format PDF (*Portable Document Format*), le format MIDI (Musical Instrument Digital Interface), le format Finale [44] et le format MusicXML [24].

5.3 Module *Analyse de la performance musicale*

Le module *Analyse de la performance musicale* est le noyau central de ce projet de recherche. Il a pour fonction de comparer le signal audio numérique avec la partition de musique afin d'extraire de l'information sur la performance musicale du musicien telle que les erreurs de performance, les segments de la partition joués et la position actuelle dans la partition. La figure 5.2 présente les différentes couches logicielles de ce module.

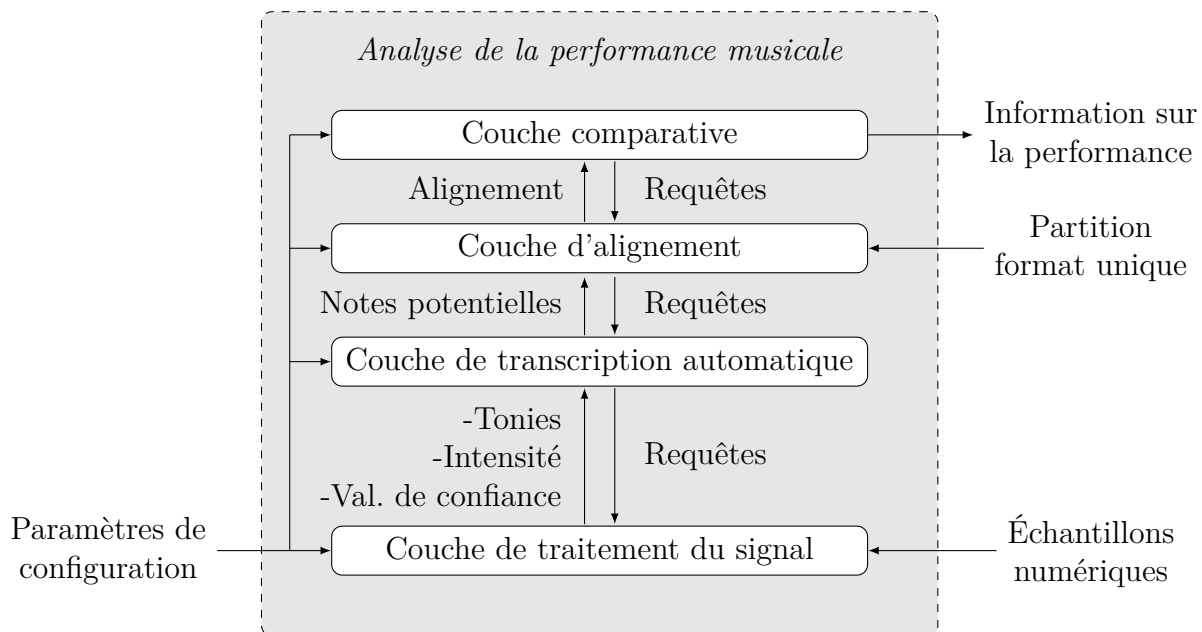


Figure 5.2 Schéma bloc global du système

Il est à noter que le fonctionnement interne de chacune de ces quatre couches est présenté en détail dans les quatre chapitres qui suivent ce chapitre respectivement. Les sous-sections suivantes résument l'utilité de chacune de ces couches.

5.3.1 Couche de traitement du signal

La couche de traitement du signal se charge d'analyser le signal audio numérique d'entrée correspondant à la performance musicale du musicien. Pour ce faire, le signal d'entrée $x[n]$ est d'abord décomposé en trames chevauchées $x[n]_i$ composées de N échantillons audios telles qu'illustrées à la figure 5.3. Les détails concernant le dimensionnement des trames et le chevauchement seront fournis à la sous-section intitulée « *Les paramètres de configuration* » de la section 6.4.1 du chapitre 6.

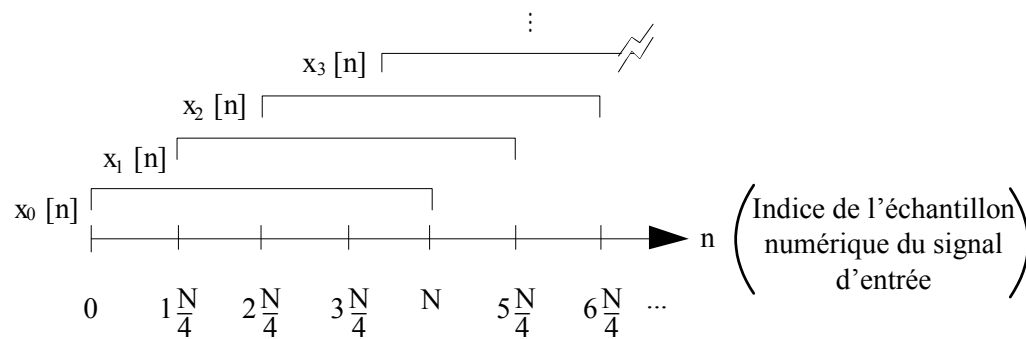


Figure 5.3 Exemple de trames chevauchées

Pour chacune de ces trames, les tonies potentielles sont estimées à l'aide d'un algorithme d'estimation polyphonique de tonies basé sur le domaine fréquentiel. Également, la couche évalue une intensité perceptuelle et une valeur de confiance qu'elle associe à chacune des valeurs de tonies estimées. À titre d'exemple, la figure 5.4 illustre les tonies estimées à partir d'un échantillon audio de violon composé des six notes de musique de la figure 5.5. Il est à noter que le niveau de gris de chacune des tonies de la figure 5.4, présentée sous forme de point, reflète la valeur de confiance de cette tonie, et ce, selon le dégradé présenté à droite de la figure.

En bref, l'objectif de cette couche consiste essentiellement à estimer toutes les tonies contenues dans le signal audio, et ce, au risque d'avoir plusieurs tonies estimées en extra.

5.3.2 Couche de transcription automatique

Pour sa part, la couche de transcription automatique a pour objectif d'estimer toutes les notes potentiellement jouées par le musicien, et ce, en utilisant comme entrée les tonies estimées par la couche inférieure de traitement du signal. Pour ce faire, la couche observe la corrélation entre les tonies adjacentes pour joindre ensemble les tonies de valeur similaire. Il est à noter que cette couche tient compte de la présence de tonies estimées en extra et manquantes. La figure 5.6 illustre un exemple des notes estimées à partir des tonies de la figure 5.4.

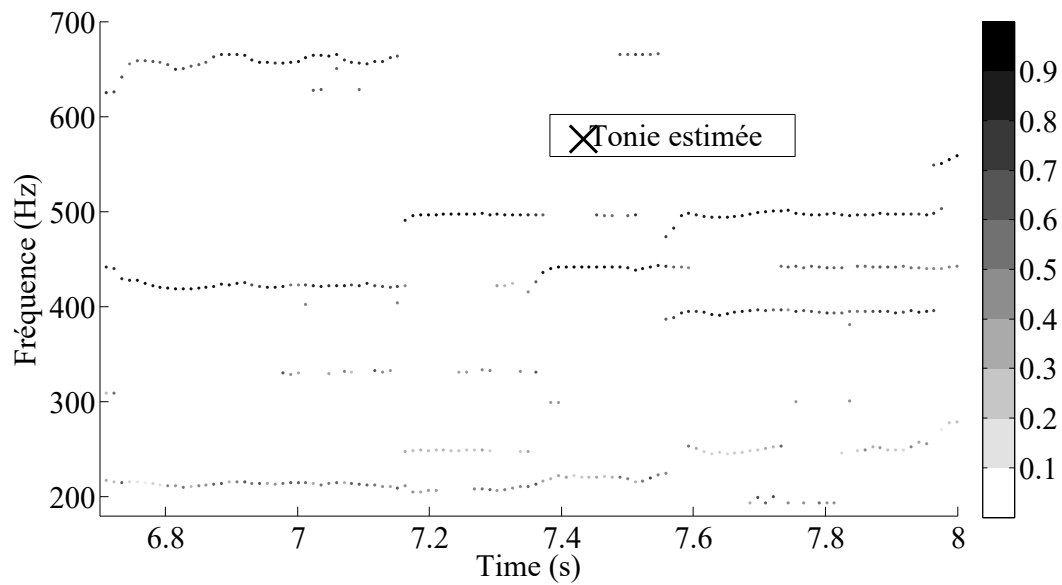


Figure 5.4 Exemple de tonies estimées par la couche de traitement du signal

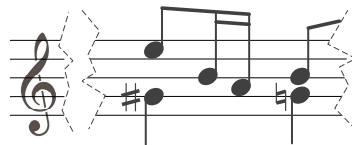


Figure 5.5 Notes de musiques jouées pour l'échantillon audio de la figure 5.4

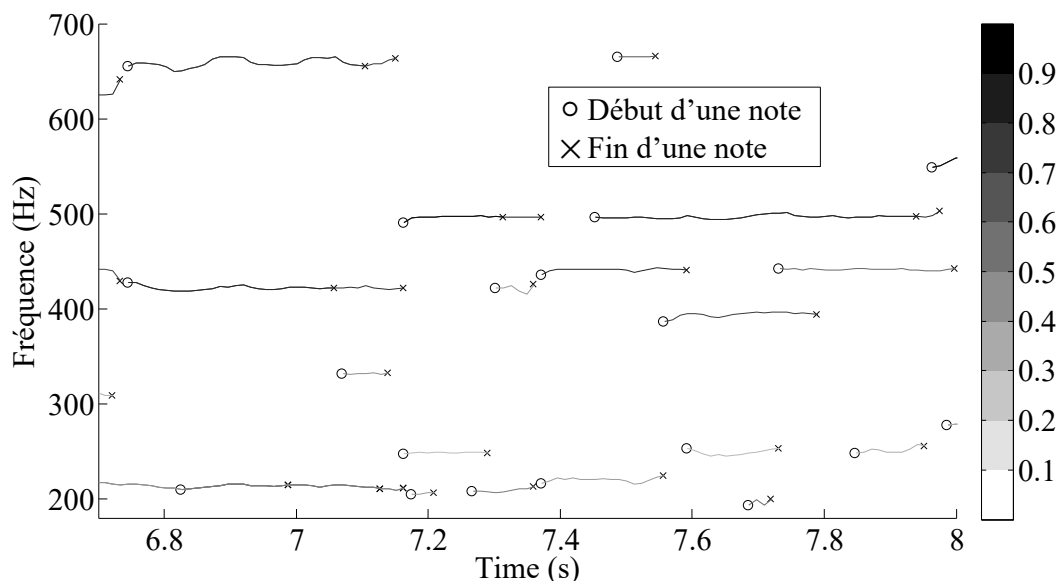


Figure 5.6 Cette figure présente un exemple de notes estimées par la couche de transcription automatique, et ce, à partir des tonies de la figure 5.4. L'intensité d'une note correspond à la moyenne des valeurs de confiance des tonies utilisées pour former la note.

Tout comme pour la couche de traitement du signal, la couche de transcription automatique a pour objectif d'estimer toutes les notes jouées par le musicien, et ce, au risque d'avoir plusieurs notes estimées en extra. C'est d'ailleurs le cas de la figure 5.6 puisque plus d'une dizaine de notes sont estimées alors que le segment audio en contenait que six.

5.3.3 Couche d'alignement

La couche d'alignement permet d'associer adéquatement les notes de la partition de musique aux notes correctement estimées par la couche de transcription automatique. Pour ce faire, la couche recherche les ressemblances sur le plan musical telles que la ressemblance entre le moment de l'attaque d'une note par rapport au moment de l'attaque attendu selon la partition de musique. Évidemment, pour être associées ensemble, chacune des notes estimées doit posséder une fréquence moyenne équivalente à la fréquence nominale de la note de la partition à laquelle elle est associée. La figure 5.7 illustre un exemple d'alignement entre certaines des notes estimées de la figure 5.6 avec les six notes de musique de la figure 5.5.

Il est à noter que la couche d'alignement réalisée donne la liberté au musicien de jouer la partition de musique à la vitesse qu'il désire, et ce, tout en lui donnant la liberté d'accélérer ou de décélérer. Cette couche donne aussi la liberté au musicien de jouer n'importe quel segment de la partition de musique sans avoir à préalablement informer le système de

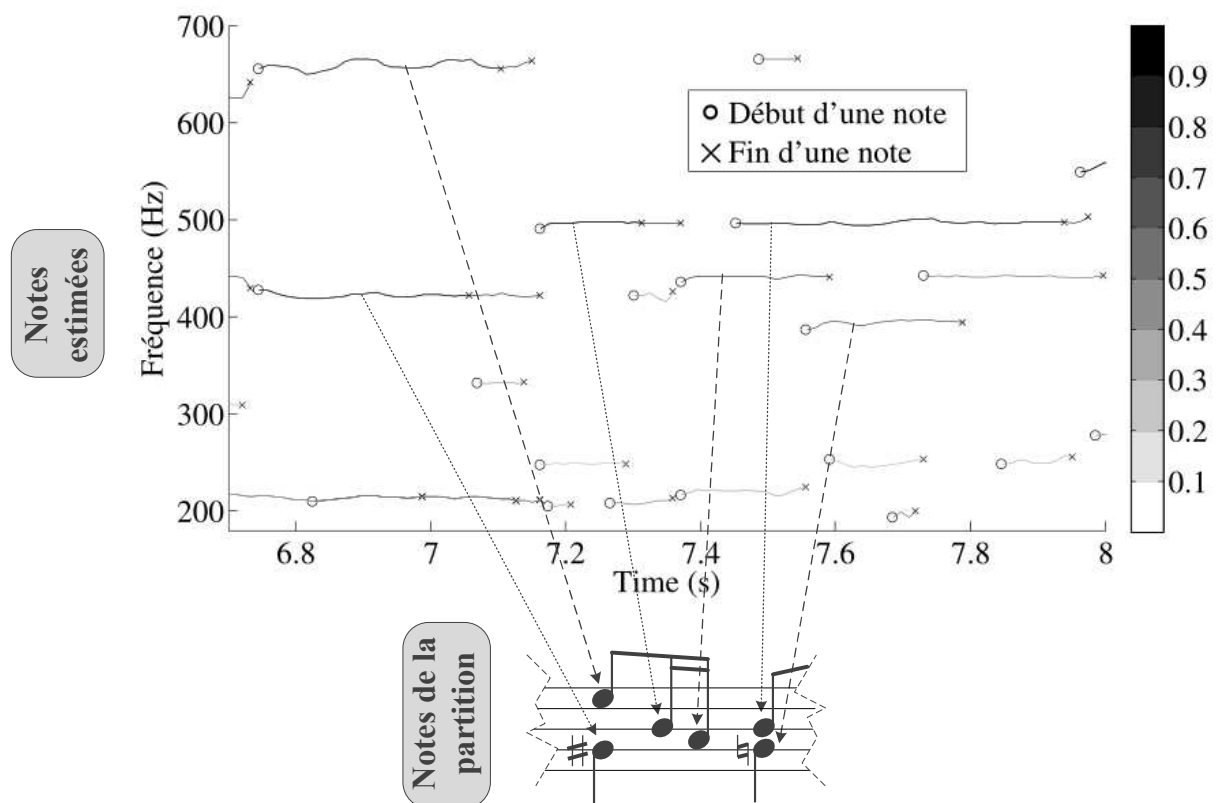


Figure 5.7 Exemple d'alignement estimé par la couche d'alignement, et ce, à partir des notes de la partition de la figure 5.5 et des notes estimées de la figure 5.6

ses intentions. Ainsi, la couche d'alignement a la capacité de retrouver ce qui est joué par le musicien partout à travers la partition de musique, et ce, pourvu que celui-ci joue suffisamment de notes. Cette liberté permet à l'interprète de se concentrer sur l'exécution de la performance musicale puisque c'est l'application qui s'adapte à son rythme et non le contraire. D'ailleurs, tel qu'il sera discuté au chapitre 8, cette particularité est l'une des contributions scientifiques importantes de ce projet de recherche.

5.3.4 Couche comparative

Cette couche est la dernière couche du module *Analyse de la performance musicale* de la figure 5.1. Cette couche permet de générer la sortie du module, c'est-à-dire une liste de résultats de comparaison entre la performance musicale du musicien et la partition de musique. Pour ce faire, la couche compare, entre autres, chacune des notes correctement estimées par la couche de transcription automatique à la note de la partition de musique à laquelle elle a été associée par la couche d'alignement. Les comparaisons effectuées par la couche permettent d'obtenir les résultats énumérés ci-dessous :

1. La précision du moment d'attaque des notes jouées par le musicien
2. La précision de la durée des notes jouées par le musicien
3. La précision de la tonie des notes jouées par le musicien

Bien que la position du musicien dans la partition de musique à un instant donné soit trouvée par la couche d'alignement, cette information est redirigée vers la sortie de la couche comparative et fait ainsi partie des diverses informations de sortie de la couche comparative.

5.3.5 Architecture de type *bottom-up*

Le module *Analyse de la performance musicale* a été conçu selon une architecture de type *bottom-up* où le niveau d'abstraction augmente du bas vers le haut, la couche de traitement du signal étant la moins abstraite et la couche comparative étant la plus abstraite. De ce fait, plus le niveau d'abstraction augmente, plus le niveau de détail diminue et plus la vue d'ensemble s'élève :

- La couche de traitement du signal traitant la moins abstraite et la plus nombreuse des entités, soit les échantillons audio.
 - La couche de transcription automatique traitant une entité plus abstraite et moins nombreuse que la couche précédente, soit les tonies estimées.
 - La couche d'alignement traitant une entité plus abstraite et moins nombreuse que la couche précédente, soit les notes estimées.
-

- La couche comparative traitant la plus abstraite des entités, soit l’alignement.

Plus particulièrement, cette architecture permet de confier aux couches supérieures le pouvoir décisionnel quant à l’utilisation ou non de l’information qui lui est fournie par la couche inférieure, et ce, comme ces dernières disposent d’une vue d’ensemble plus globale leur permettant de prendre des décisions plus éclairées et potentiellement plus judicieuses.

5.4 Module *Application*

Le module *Application* peut varier, par exemple, il est possible que ce module réalise un jeu similaire à *Guitar Hero* ou bien un professeur de musique virtuel de violon ou tout simplement un logiciel permettant de tourner automatiquement les pages d’une partition de musique tel que le logiciel Tonara. Pour ce projet de recherche, ce module a été conçu pour permettre essentiellement l’affichage texte des diverses informations de sorties du module *Analyse de la performance musicale*, et ce, dans une interface graphique de tests et de débogage. Cette interface permet aussi de charger les diverses entrées du système telles que la partition de musique et les différents paramètres de configuration. Les quatre prochains chapitres présentent les détails de fonctionnement des différentes couches du module *Analyse de la performance musicale*.

CHAPITRE 6

COUCHE DE TRAITEMENT DU SIGNAL

Ce chapitre présente le fonctionnement de la couche de traitement du signal utilisée par le module *Analyse de la performance musicale* de la figure 5.2. La couche de traitement du signal utilise comme entrée différentes variables de configuration et le signal audio numérique provenant du module *Saisie du signal audio* de la figure 5.2, c'est-à-dire le signal correspondant à la performance musicale du musicien. La couche décompose d'abord le signal numérique $x[n]$ en trames chevauchées $x_i[n]$ telles qu'illustrées à la figure 6.1.

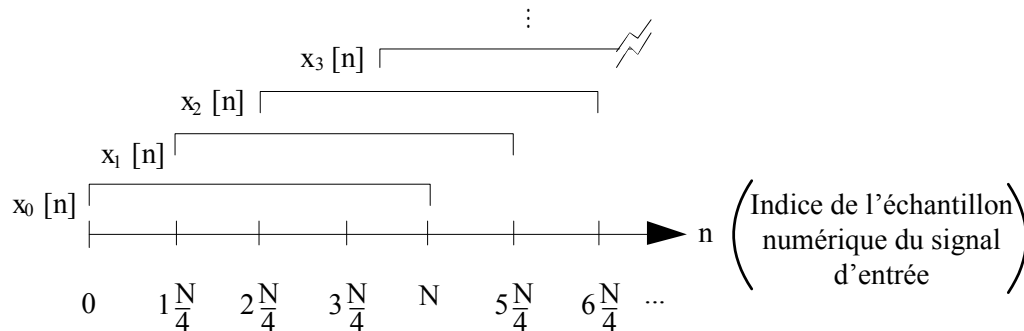


Figure 6.1 Exemple de trames chevauchées

La décomposition en trames permet l'analyse fréquentielle d'un signal non stationnaire tel qu'un signal audio de musique composé de plusieurs notes. En effet, cette décomposition permet de créer des petits segments audio (trames) dont les statistiques sont suffisamment stationnaires pour permettre l'analyse fréquentielle. Pour sa part, le chevauchement permet d'augmenter la résolution temporelle en rapprochant les trames les unes des autres dans le temps.

Pour chacune de ces trames, le rôle de la couche consiste essentiellement à estimer les tonies potentielles exprimées en hertz (Hz) telles qu'illustrées à la figure 6.2 pour un extrait de signal de violon composé de deux notes consécutives. La couche a aussi pour fonction d'associer deux attributs à chacune des tonies estimées, c'est-à-dire une intensité basée sur la psychoacoustique ainsi qu'une valeur de confiance. Les diverses tonies estimées accompagnées de leurs attributs forment la sortie de la couche. Ces données permettent à la couche supérieure, c'est-à-dire la couche de transcription automatique, d'estimer les notes potentiellement jouées par le musicien.

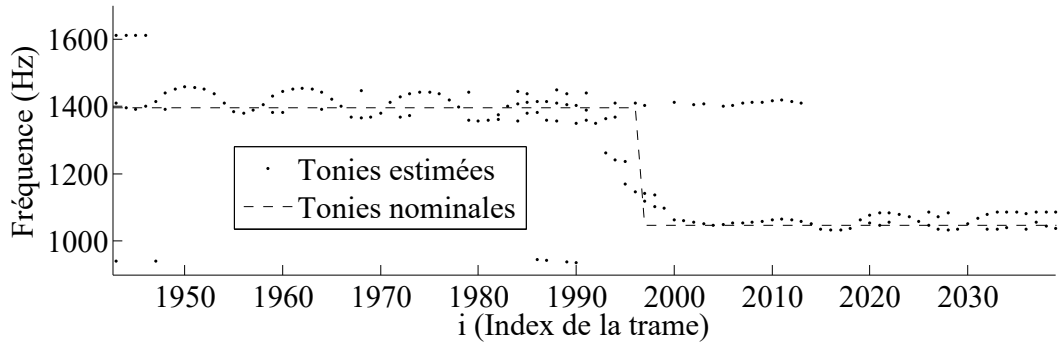


Figure 6.2 Exemple de tonies estimées par la couche traitement du signal

La couche de traitement du signal est basée sur une approche de type *bottom-up* puisqu'elle fournit plus de tonies que nécessaire à la couche supérieure de transcription automatique. En fait, la couche de traitement du signal a pour objectif d'estimer toutes les tonies contenues dans le signal, et ce, malgré l'estimation de quelques tonies en extra. Donc, plutôt que d'éliminer des tonies incertaines, la couche de traitement du signal transmet ces dernières à la couche supérieure. Ainsi, la couche de transcription automatique est le niveau où le système proposé prend la décision d'utiliser ou non les tonies qui lui ont été transmises.

En fait, la couche supérieure de transcription automatique n'a pas nécessairement besoin que toutes les tonies estimées soient fiables à 100% puisqu'il ne faut pas perdre de vue que cette couche possède un niveau d'abstraction supérieur. D'abord, cette dernière fonctionne avec des tonies comparativement à la couche de traitement du signal qui fonctionne avec des échantillons audio. Ensuite, tel qu'il est possible d'observer pour la figure 6.2, il existe normalement une forte corrélation entre les tonies adjacentes appartenant à une note jouée. Ainsi, comparativement à la couche de traitement du signal, la couche de transcription automatique peut se baser sur cette corrélation pour décider de rejeter ou non une tonie. En effet, une tonie isolée est beaucoup moins probable qu'une tonie entourée de plusieurs autres tonies similaires. De plus, tel qu'il peut être observé à partir de la figure 6.2, la présence d'un vibrato peut permettre le rejet des tonies qui peuvent sembler incohérentes avec ce dernier.

Cela dit, la couche de transcription automatique et la couche de traitement du signal ont été conçues pour fonctionner en temps réel en présence de notes rapides et de notes jouées en polyphonie, et ce, dans le but d'estimer les notes jouées par le musicien afin de pouvoir éventuellement répondre à la question de recherche exposée initialement à l'introduction.

Il est à noter que la couche de traitement du signal a été conçue pour fonctionner en présence de pièces de violon de calibre professionnel contenant des segments lents et rapides joués dans des styles variés tels que le legato et le staccato. Les prochaines sections présentent les détails liés à cet instrument de musique, et ce, suivi du fonctionnement détaillé de la couche de traitement du signal.

6.1 Instrument de musique visé par la couche

Bien que cette couche soit spécialisée pour le violon, les couches supérieures illustrées à la figure 5.2 ont été conçues pour faire abstraction du type d'instrument traité par cette couche. En d'autres mots, pour que le système proposé fonctionne avec un autre instrument de musique, il suffit de modifier le fonctionnement interne de la couche de traitement du signal, et ce, tout en conservant l'interface entre les deux couches, c'est-à-dire l'interface entre la couche de traitement du signal et la couche de transcription automatique.

Il est à noter que le violon a été choisi pour diverses raisons. D'abord, un violoniste peut jouer des notes très rapidement, avec ou sans un vibrato, et ce, avec des styles variés tels que le legato, le staccato, le pizzicato et le détaché. Ensuite, comme l'estimation de tonies contenues dans un signal polyphonique est encore aujourd'hui un défi complexe à relever, il est jugé sage de travailler avec un instrument légèrement polyphonique tel que le violon, et ce, comparativement à un instrument hautement polyphonique tel que le piano. En effet, le violon est considéré comme légèrement polyphonique puisqu'il permet de jouer un maximum de quatre notes en simultanée.

Afin de s'assurer que la couche de traitement du signal conçue soit performante pour le violon, il a d'abord été nécessaire d'analyser la nature des signaux audio produits par le violon. Cette sous-section présente les résultats de cette analyse.

6.1.1 Inharmonicité des signaux de violon

Le violon génère des notes dont la tonie peut varier de 196 Hz à environ 2000 Hz. Puisque le violon est un instrument à cordes, il génère aussi des signaux inharmoniques [35]. Les signaux générés sont principalement constitués de fréquences fondamentales (f_0) avec leurs harmoniques correspondantes. Quand une corde d'un instrument vibre librement après une excitation, les fréquences harmoniques ne sont pas précisément égales à un multiple entier de la fréquence fondamentale. En fait, les fréquences d'harmoniques sont progressivement décalées vers la droite de l'axe des fréquences. Cette inharmonicité est causée par le manque d'élasticité de la corde. Ainsi, les cordes courtes, épaisses et rigides présenteront plus d'inharmonicité [19]. La figure 6.3 illustre un exemple d'inharmonicité où le décalage

progressif vers la droite des harmoniques est perceptible vers 15 KHz pour la corde ouverte MI du violon ($f_0=659$ Hz) jouée en style pizzicato, en pinçant la corde avec un doigt.

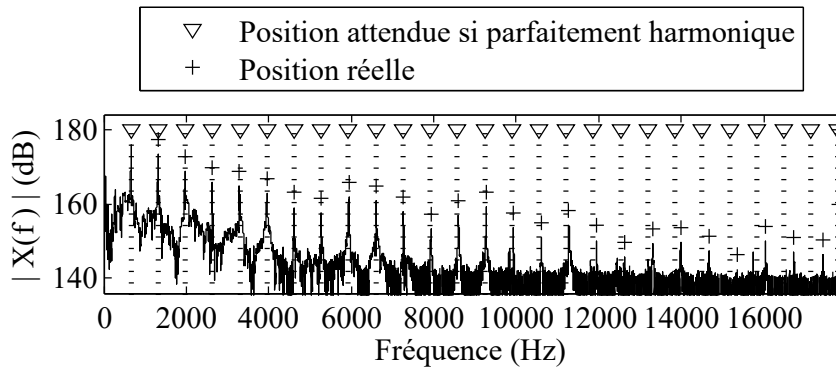


Figure 6.3 Spectre en fréquence de la corde ouverte MI (659.5 Hz) jouée en pizzicato

La figure 6.4 illustre le module en fréquence pour la même note de violon que précédemment, mais cette fois jouée en frottant l'archet sur la corde. Pour ce cas, il est possible de remarquer l'absence d'inharmonicité causée par le fait que la corde ne vibre pas librement comme elle est continuellement frottée par l'archet. Ce phénomène physique, appelé verrouillage de mode, fait que les harmoniques se verrouillent exactement à des multiples entiers de la fréquence fondamentale [19].

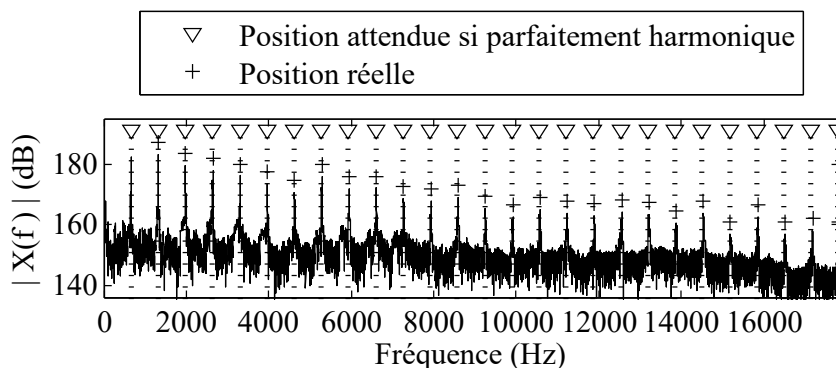


Figure 6.4 Spectre en fréquence de la corde ouverte MI (659.5 Hz) jouée avec l'archet

L'inharmonicité ne peut pas être complètement ignorée puisque les pièces de violon peuvent contenir des notes jouées en pizzicato et en staccato. À titre indicatif, une note jouée en staccato est produite en utilisant d'abord l'archet pour exciter la corde et, ensuite, en permettant à la corde de vibrer librement. Cependant, si seulement quelques harmoniques (5 ou moins) sont utilisées pour estimer la tonie, l'inharmonicité peut être négligée tel que

démontré par les résultats du tableau 6.1. Il est à noter que les résultats de ce tableau ont été obtenus en évaluant l'inharmonicité des notes de violon provenant de la notoire base de données audio RWC [25], soit, plus précisément, les notes de violon jouées seules contenues dans le volume numéro 15 pour violon. De même, les figures 6.3 et 6.4 ont aussi été créées à partir des notes de ce volume.

Tableau 6.1 Pourcentage de déviation de la fréquence de l'harmonique par rapport à la fréquence attendue

Fréquence fondamentale de la note	Index de l'harmonique					
	1	2	3	4	5	6
	Pourcentage de déviation					
196 Hz (archet)	0.5	0.7	0.6	1.0	1.3	3.5
1568 Hz (archet)	0.2	0.1	0.4	0.2	0.5	1.0
196 Hz (pizz.)	0.3	4.2	1.0	1.5	2.9	2.8
1559 Hz (pizz.)	0.2	0.6	0.2	0.8	0.5	0.2

6.1.2 Polyphonie des signaux de violon

Le violon est un instrument polyphonique puisqu'il possède quatre cordes indépendantes pouvant vibrer simultanément. De plus, avec l'archet, deux cordes peuvent être frottées en simultanée. En fait, il existe plusieurs pièces de violon bien connues contenant des segments de musique polyphoniques où quatre notes doivent être jouées en simultanée en frottant simultanément les deux cordes basses et, juste ensuite, les deux cordes hautes du violon. Bien qu'en réalité, les quatre notes ne soient pas jouées en simultanée par le violoniste, les deux cordes basses continuent tout de même à vibrer pendant que les deux cordes hautes sont frottées à leur tour.

De plus, une partition de musique ne contenant que des notes monophoniques pourrait aussi générer un signal polyphonique en raison de la résonance des notes jouées précédemment sur les autres cordes. D'ailleurs, même en utilisant un bon estimateur monophonique de tonies, la présence de notes précédentes rend parfois difficile l'estimation de tonies, et ce, plus particulièrement en présence de segments rapides puisque les notes sont rapprochées dans le temps et, par conséquent, certaines n'ont pas le temps de s'estomper complètement avant le jeu des notes suivantes. Par exemple, même en utilisant l'algorithme monophonique YIN [9], la dernière note de quatre notes staccato rapides [59] n'a pas été correctement estimée à cause de la résonance comme l'illustre la figure 6.5. Par conséquent, afin d'estimer efficacement les tonies contenues dans des pièces rapides de violon, il est essentiel d'utiliser un estimateur polyphonique de tonies.

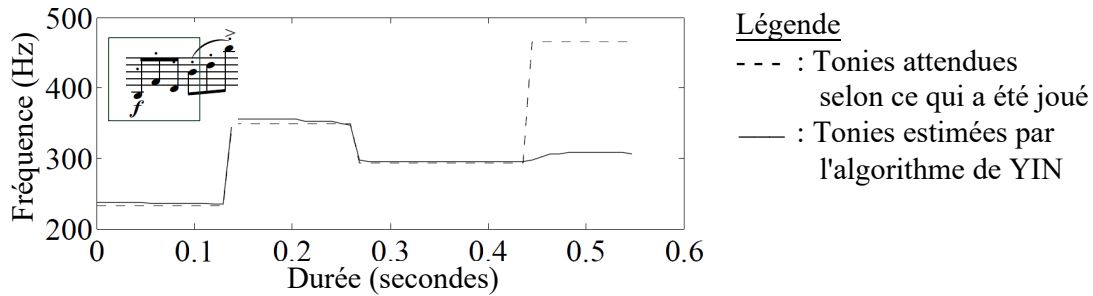


Figure 6.5 Résultat d'estimation monophonique pour quatre notes rapides jouées en staccato où la tonie de la dernière note n'a pas été estimée correctement

6.2 Fichiers audio utilisés et critères d'évaluation

Le violon est un instrument intéressant puisque les violonistes peuvent jouer à haute vitesse (jusqu'à 15 notes par seconde dans la pièce de la référence [81]) et peuvent jouer les notes en divers styles tels que le détaché, le legato et le staccato et peuvent modifier la tonie et l'intensité sonore d'une note comme ils le désirent. Par conséquent, pour évaluer correctement la performance d'un estimateur de tonies dédié au violon, il est important d'utiliser des signaux audio qui contiennent toutes les caractéristiques citées ci-dessus. Ainsi, les évaluations ont été réalisées à l'aide d'une pièce de violon soigneusement sélectionnée où la position temporelle de chacune des notes a été déterminée manuellement, c'est-à-dire que leur début et leur fin ont été soigneusement trouvés en écoutant de façon répétée le signal audio. Cette pièce est le deuxième mouvement de la sonate pour violon no.2 de Prokofiev joué par Gil Shaham [59]. Cette pièce est composée d'un thème (nommé, *Tema*) et de cinq variations. La table 6.2 résume tous les segments musicaux contenus dans l'enregistrement.

Tableau 6.2 Caractéristiques de l'enregistrement audio

Segment de musique / polyphonie	Caractéristiques des notes		
	Nb. de notes	Durée moyenne	Attaque typique
Tema (mono)	40	0.90 s.	Legato
Var. I (mono)	83	0.37 s.	Legato
Var. II (mono)	109	0.20 s.	Staccato
Var. III (mono)	45	0.87 s.	Legato
Var. IV (mono)	175	0.14 s.	Legato
Var. V (duo)	77	0.56 s.	Détaché

Essentiellement, cet enregistrement a été sélectionné pour les raisons suivantes :

1. Il est facilement accessible et, de ce fait, il permet à d'autres chercheurs de comparer leurs résultats à ceux présentés dans ce document.
2. Il s'agit d'un enregistrement professionnel.
3. Le violon est joué en solo (sans accompagnement).
4. Il peut être segmenté en quelques jours de travail intensif.
5. Les notes sont jouées à différente vitesse avec différents styles.
6. Il contient des notes jouées en polyphonie.

Principalement, deux critères seront utilisés pour évaluer la performance : le pourcentage de tonies estimées correctement et le pourcentage de tones estimées en erreur. Il est à noter qu'une seule tonie peut être associée à une tonie nominale, les doublons seront considérés comme des tonies en extra. Pour être considérée comme identique, la tonie estimée doit avoir une distance de moins de ± 75 cents par rapport à la tonie de la note nominale où la distance en cents entre deux tonies (f_1 et f_2) est calculée en utilisant l'équation 6.1.

$$cents = 1200 \log_2(f_2/f_1) \quad (6.1)$$

Tel qu'illustré à la figure 6.6, seulement 90% de la durée des notes nominales sera utilisée pour le calcul des deux critères de performance, c'est-à-dire que les tonies estimées dans la zone en pointillés ne seront pas comptabilisées. Cette décision est basée sur le fait que la position nominale de début et de fin des notes a été trouvée manuellement en écoutant le signal. Par conséquent, ils sont enclins à une petite erreur humaine. Il est à noter que l'estimateur de tonies proposé fonctionne bien lors des transitions de notes et, dans les faits, les résultats obtenus seraient très similaires sans cette tolérance.

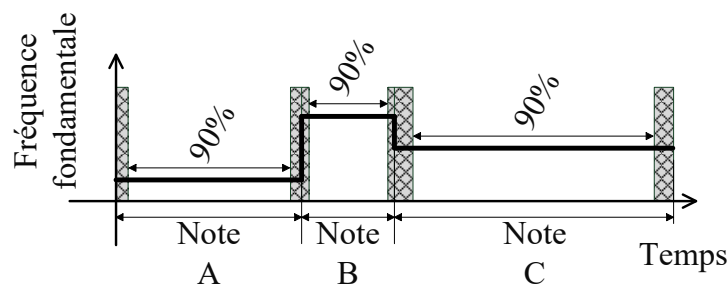


Figure 6.6 Portion du signal utilisé pour l'évaluation de la performance

6.3 Algorithme de base utilisé pour réaliser la couche

Camacho et Harris ont passé en revue plusieurs algorithmes potentiels pour estimer les tonies contenues dans un signal harmonique [6]. L'algorithme décrit par l'équation 6.2 et

présenté dans la référence [6] a été choisi comme base pour l'estimateur de tonies proposé dans ce projet de recherche, où p et $X(f)$ représentent la tonie et le spectre de fréquence respectivement.

$$p = \arg \max_f \int_0^\infty |X(f')| \sum_{n=1}^N \delta(f' - nf) df' \quad (6.2)$$

L'équation 6.2 estime la tonie en trouvant la fréquence donnant la somme maximale de l'amplitude de la fréquence correspondante (f) additionnée à l'amplitude de ses harmoniques ($nf \forall n > 1$). Pour simplifier, cette somme sera maintenant appelée la somme harmonique. Il est intéressant de noter que l'équation 6.2 n'est valide que si les harmoniques sont positionnées à un multiple entier de la fréquence fondamentale, c'est-à-dire pour un son harmonique parfait. Nous posons maintenant l'hypothèse que l'inharmonicité du violon ne fera pas échouer cet algorithme, et ce, pourvu que la variable N soit maintenue basse.

Il est à noter que d'autres chercheurs [14, 37, 84] ont aussi basé leur estimateur de tonies sur des principes semblables, c'est-à-dire en se basant sur la structure des signaux inharmoniques.

6.4 L'estimateur de tonies proposé

Les étapes pour implémenter l'estimateur de tonies proposé sont listées ci-dessous. Chacune de ces étapes sera détaillée dans les sous-sections suivantes.

1. Calculer la somme harmonique ;
2. Rechercher les tonies potentielles ;
3. Calculer le niveau de confiance des tonies ;
4. Sélectionner les meilleurs candidats.

6.4.1 Calculer la somme harmonique

La première étape consiste à trouver la somme harmonique pour chaque fréquence afin de générer un spectre de somme harmonique tel que défini par l'équation 6.3. Ce spectre de somme harmonique sera maintenant appelé le *Harmsum*.

$$H(f) = \int_0^\infty |X(f')| \sum_{n=1}^N \delta(f' - nf) df' \quad (6.3)$$

Pour des raisons pratiques, il est nécessaire d'avoir une version discrète de l'équation 6.3. La version discrète est définie par l'équation 6.4, où $X[k]$ est le résultat de la transformée de Fourier discrète (TFD) appliquée sur les échantillons $x[n]$ d'une trame de L échantillons. Dans notre cas, une fenêtre Hanning a été utilisée pour pondérer les échantillons audio $x[n]$ et la TFD a été implémentée en utilisant la transformée de Fourier rapide (FFT). De plus, pour l'équation 6.4, la partie symétrique de $X[k]$ a été supprimée, car elle contient des informations redondantes.

$$H[k] = \sum_{k'=1}^{L/2} |X[k']| \sum_{n=1}^N \delta[k' - nk] \quad (6.4)$$

La figure 6.7 illustre un exemple d'*Harmsum* ($N = 6$) appliqué sur un signal audio synthétisé correspondant à un signal harmonique parfait. Dans ce cas, l'amplitude maximale $H[5]$ correspond à la cinquième (5^e) raie d'*Harmsum*, soit celle correspondant à la fréquence fondamentale.

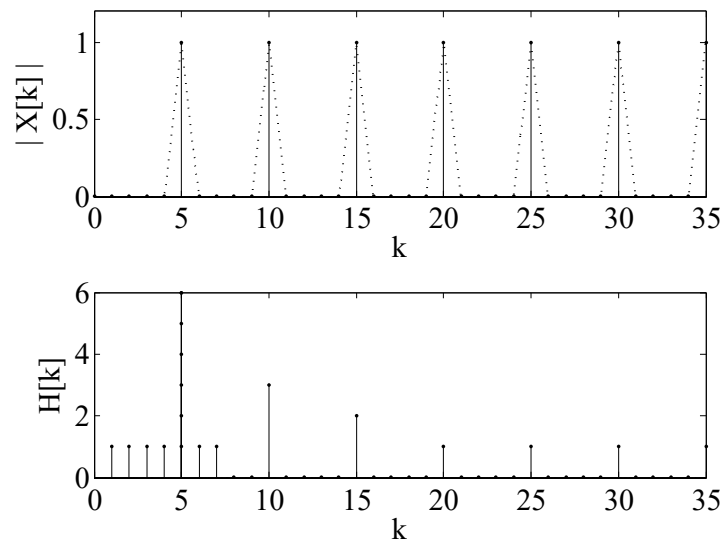


Figure 6.7 Spectre en fréquence et *Harmsum* résultant pour un signal fictif parfaitement harmonique où la fréquence fondamentale correspond à la raie d'indice 5

Importance de la résolution et de la granularité spectrale

Pour des signaux musicaux réels, les fréquences peuvent être positionnées entre deux raies d'indice k et $k + 1$ du spectre fréquentiel $X[k]$. Cela étant, il est vital que la résolution en fréquence de $X[k]$ soit suffisamment élevée pour que les algorithmes de traitement du signal puissent bien identifier la ou les fréquence(s) fondamentale(s) contenues dans les

segments de signal audio, et ce, dans le but éventuel de permettre au système d'identifier correctement la tonalité (Do4, Ré4, etc.) des diverses notes jouées.

Bien que la résolution spectrale puisse être augmentée en élevant le nombre d'échantillons audio utilisé pour calculer la FFT, il y a tout de même des limitations quant à la possibilité de déterminer avec précision la fréquence fondamentale d'une note ou, voir même, de distinguer deux fréquences rapprochées l'une de l'autre appartenant à deux notes distinctes. En fait, plus les notes sont de longue durée, plus il sera possible de distinguer avec précision leur structure fréquentielle harmonique, et ce, en utilisant un nombre élevé d'échantillons audio pour calculer la FFT et, à l'inverse, plus les notes sont de courte durée, plus il sera difficile de distinguer avec précision leur structure fréquentielle harmonique, et ce, malgré qu'un nombre élevé d'échantillons audio soit utilisé pour calculer la FFT. Concrètement, plus le support temporel sera court, tel que c'est le cas pour une note de courte durée, plus le contenu fréquentiel sera diffusé et, par conséquent, plus il sera difficile de déterminer avec précision la tonie de cette dernière [29]. En contrepartie, la résolution temporelle d'une note de courte durée étant élevée, la dimension de la FFT doit, pour sa part, être petite pour permettre l'identification précise du moment de début et de fin de la note. Il est à noter que cette dualité entre le domaine fréquentiel et le domaine temporel est liée au principe d'incertitude d'Heisenberg [45].

Ceci étant, pour estimer adéquatement les tonies contenues dans un segment de signal audio à l'aide de l'équation 6.4, non seulement la résolution fréquentielle doit être suffisamment élevée, mais la granularité du spectre $X[k]$ (nombre de raies spectrales) doit aussi être suffisamment élevée. Or, il est possible d'augmenter la granularité du spectre $X[k]$, et ce, sans affecter la résolution fréquentielle et temporelle, en ajoutant des échantillons nuls à la fin de la trame d'échantillons audio avant le calcul de la FFT (*zero padding*, en anglais).

Pour illustrer l'importance de la granularité du spectre $X[k]$, les figures 6.8 et 6.9 illustrent deux exemples où la fréquence fondamentale nominale est comprise entre les raies d'indice 5 et 6 du spectre fréquentiel, c'est-à-dire à la raie spectrale 5.2 et 5.3 respectivement.

Pour le cas de la figure 6.8, l'amplitude maximale du *Harmsum* correspond toujours à raie la plus proche de la fréquence fondamentale ($X[5]$), mais l'amplitude est maintenant 3 fois plus faible que pour le cas de la figure 6.7, et ce, malgré que la puissance soit équivalente. Maintenant, pour l'exemple de la figure 6.9, l'amplitude maximale de l'*Harmsum* ne correspond même plus à la raie la plus proche de la fréquence fondamentale. Pour ce cas, l'amplitude maximale d'*Harmsum* est maintenant liée aux raies $X[8]$ et $X[16]$.

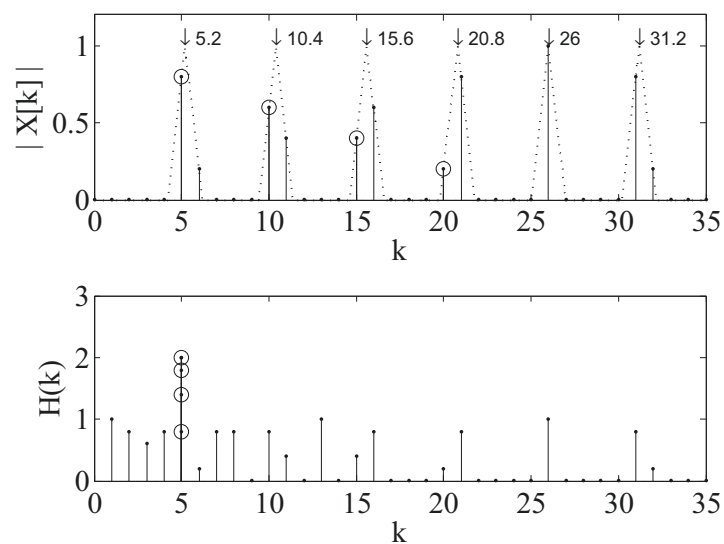


Figure 6.8 Spectre en fréquence et *Harmsum* résultant pour un signal fictif parfaitement harmonique où la fréquence fondamentale correspond à la raie d'indice 5.2

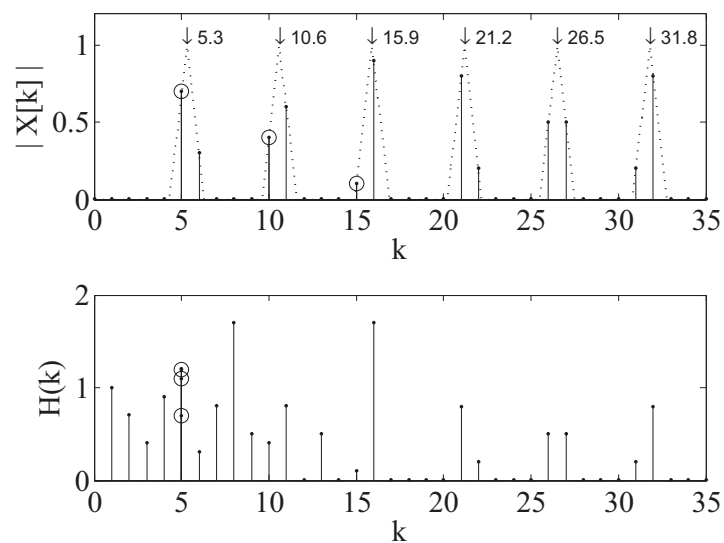


Figure 6.9 Spectre en fréquence et *Harmsum* résultant pour un signal fictif parfaitement harmonique où la fréquence fondamentale correspond à la raie d'indice 5.3

La sous-section suivante présente la démarche utilisée pour déterminer les paramètres qui seront utilisés pour la couche de traitement du signal, tels que la quantité d'échantillons audio par trame et la quantité d'échantillons nuls à ajouter à la fin de chacune de ces trames.

Les paramètres de configuration

Tel que mentionné à la référence [6], le module du spectre fréquentiel $|X[k']|$ peut être modifié avant d'appliquer l'équation 6.4. Par exemple, il est possible de modifier chacune des composantes du module $|X[k']|$ par une opération logarithmique pour écraser le spectre en fréquence afin que les valeurs d'amplitude des harmoniques soient plus rapprochées entre elles. Le tableau 6.3 présente les meilleurs résultats obtenus pour différentes variantes en utilisant les paramètres listés dans le tableau 6.4, une fréquence d'échantillonnage de 44,1 KHz et, finalement, les fichiers audio et critères d'évaluation présentés à la section 6.2. Il est à noter que, pour le tableau 6.4, la longueur TFD correspond à la somme du nombre d'échantillons contenu dans la trame et d'échantillons à valeur nulle ajoutés à la suite de la trame, soit le remplissage nul. De plus, pour la table 6.3, deux tonies ont été estimées pour chacune des trames d'échantillons en utilisant la technique d'estimation de tonies décrite par l'organigramme de la figure 6.12 qui sera présentée dans la sous-section suivante 6.4.2.

Tableau 6.3 Performance et meilleure combinaison de paramètres obtenue pour diverses transformées suite à la TFD

Var.	$ X[k'] $ remplacé par	Meilleurs paramètres		% de tonies estimées correctement
		Longueur de la trame	N	
1	$ X[k'] $	2048/4096	5	89.9%
2	$\log_{10} X[k'] $	4096/4096	2	82.6%
3	$ X[k'] ^2$	2048/4096	5	85.7%
4	$ X[k'] ^{\frac{1}{2}}$	2048/4096	5	92.2%

Tableau 6.4 Paramètres testés

Longueur de la trame	256, 512, 1024, 2048 et 4096
Longueur TFD	256, 512, 1024, 2048 et 4096
N (equation 6.4)	1, 2, 3, 4, 5 et 6

Selon le tableau 6.3, la variante 4 a permis d'obtenir la meilleure performance. Ainsi, pour le calcul du *Harmsum*, la racine carrée sera d'abord appliquée à chacun des modules de fréquence $|X[k']|$, et ce, tel que décrit par l'équation 6.5.

$$H^*[k] = \sum_{k'=1}^{L/2} \sqrt{|X[k']|} \sum_{n=1}^N \delta[k' - nk] \quad (6.5)$$

Pour la suite de ce document, à moins d'être spécifié autrement, les paramètres de la variante 4 seront utilisés, soit une trame de 2048 échantillons, une longueur TFD de 4096, un saut constant entre les trames de 512 échantillons et une valeur de $N = 5$. Il est à noter que ces paramètres sont le résultat d'une recherche expérimentale, de type boîte noire, où toutes les combinaisons possibles de paramètres du tableau 6.4 ont été testés pour chacune des variantes du tableau 6.3. Cette démarche a été préconisée pour sa simplicité de conception et de mise en œuvre comme elle permet de définir un ensemble générique de paramètres pour l'ensemble de tous les signaux audio de violon analysés.

6.4.2 Rechercher les tonies potentielles

Comme le montre la figure 6.10, même avec les meilleurs paramètres, certaines tonies ne sont toujours pas correctement estimées lorsque deux tonies par trames sont estimées. Ceci est généralement attribué aux erreurs d'octave. Une erreur d'octave se produit lorsque la fréquence fondamentale estimée est un multiple entier de la fréquence fondamentale nominale.

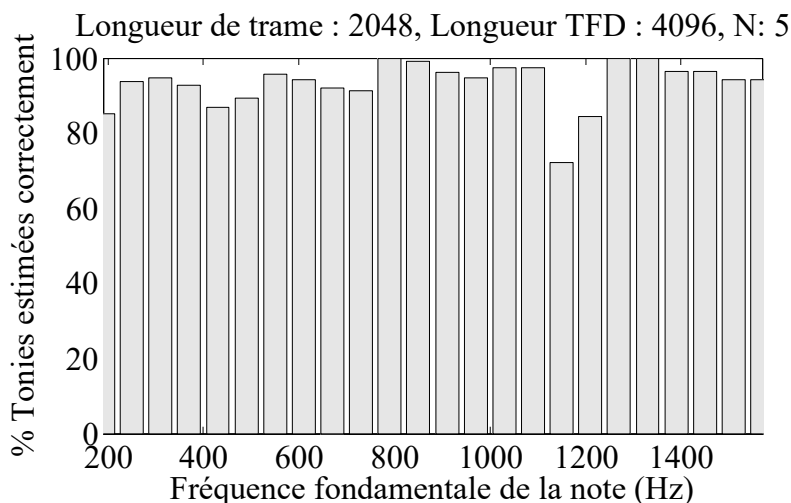


Figure 6.10 Performance pour différentes tonies en utilisant les meilleurs paramètres

L'augmentation du nombre de tonies estimées par trame aidera à augmenter le pourcentage de tonies estimées correctement, mais il est clair que cela augmentera également la quantité de tonies estimées en erreur. Néanmoins, il sera supposé pour la suite que les tonies en erreur seront facilement éliminées lors des prochaines étapes de traitement.

La figure 6.11 présente le pourcentage de tonies estimées correctement pour différentes quantité de tonies estimées par trame. Ces résultats sont comparés à notre implémentation de l'estimateur de tonies pour violon décrit à la référence [31], soit le meilleur estimateur de tonies pour violon trouvé dans la littérature scientifique. Afin d'éviter que les résultats

de comparaison de la figure 6.11 soient biaisés par une sélection défavorable de paramètres employés pour configurer l'estimateur de tonie de référence, ces derniers paramètres ont été sélectionnés à l'aide d'une démarche empirique de type essai-erreur ayant comme objectif la maximisation de la performance de l'estimateur de référence¹. Or, à partir de la figure 6.11, il est clair que l'estimateur de tonies proposé est supérieur à cet estimateur de tonies qui est lui-aussi optimisé pour le violon.

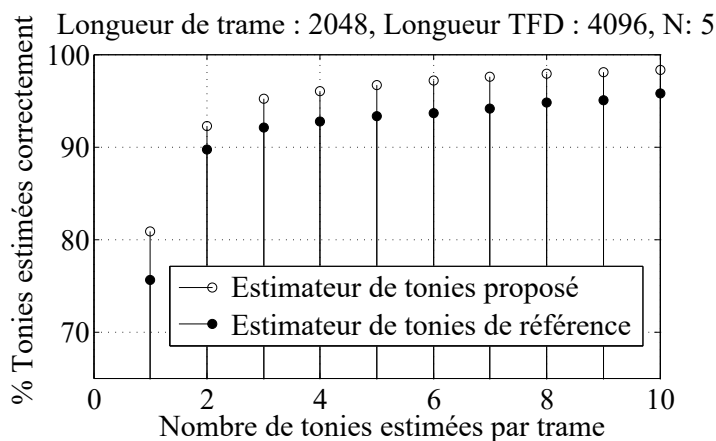


Figure 6.11 Performance en polyphonie de l'estimateur de tonies proposé pour le violon comparé à l'estimateur de référence [31]

La méthode proposée pour estimer toutes les tonies potentielles est illustrée selon l'organigramme de la figure 6.12 où $H[k]$ est le *Harmsum* de K raies, f_s est la fréquence d'échantillonnage, L est le nombre de raies spectrales de la TFD et Q est le nombre désiré de tonies potentielles p .

Tout d'abord, la raie d'amplitude maximale du *Harmsum* est trouvée. La fréquence correspondant à cette raie est enregistrée comme une tonie potentielle p . Ensuite, toutes les raies d'*Harmsum* correspondant au pic de cette fréquence sont mises à 0 où le début et la fin du pic sont le point d'inflexion gauche et le point d'inflexion droit. Finalement, ces étapes sont répétées jusqu'à ce que le nombre désiré de tonies potentielles Q soit atteint. Dans ce cas, Q est ajusté à 4, car le violon peut produire un maximum de 4 tonies en simultanée.

De plus, pour augmenter davantage les performances de l'algorithme, les raies de l'*Harmsum* correspondant aux tonies qui sont à l'extérieur de la plage attendue du violon sont simplement mises à zéro avant l'estimation des tonies potentielles.

1. Les paramètres de configuration obtenus suite à la démarche correspondent à une longueur de trame de 512 échantillons, une longueur TFD de 4096 échantillons, α défini à 5, une réduction à un cinquième et une valeur de coefficient de corrélation inverse de 0,02. Il est à noter que la signification de chacune de ces variables peut être retrouvée à la référence [31]

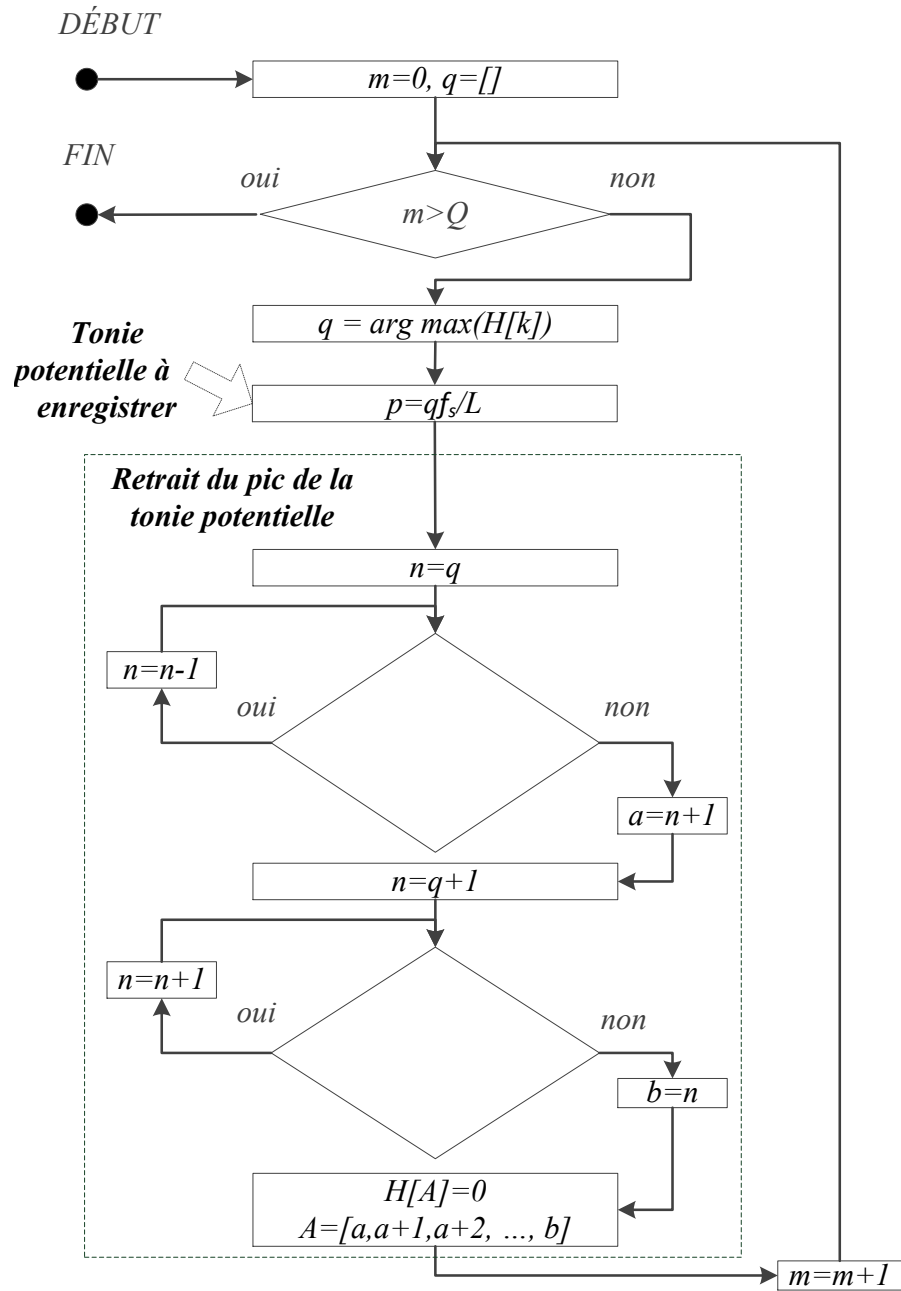


Figure 6.12 Organigramme utilisé pour estimer les tonies

6.4.3 Calculer le niveau de confiance des tonies

Comme le violon génère des signaux polyphoniques, le spectre en fréquence peut être composé de pics multiples où certains des pics peuvent correspondre à la superposition de diverses harmoniques ou fréquences fondamentales. Cette complexité spectrale peut produire des pics dans l'*Harmsum* pouvant ne correspondre à aucune tonie en particulier.

Afin de supprimer ces tonies estimées en erreur, chaque candidat potentiel p est pondéré en effectuant des observations dans le spectre fréquentiel $X[k]$. Chaque observation i est réalisée en utilisant une fonction $c_i(p, X)$ donnant un niveau de confiance dont la valeur est comprise dans l'intervalle $[0, 1]$. Finalement, les niveaux de confiance associés à chacune des observations sont combinés ensemble en utilisant la fonction moyenne selon l'équation 6.6. Il est à noter que la nature des observations réalisés dans le spectre fréquentiel ainsi que les fonctions $c_i(p, X)$ utilisées pour cela sont décrites à la section 6.5.

$$c(p) = \frac{1}{N} \sum_{i=1}^N c_i(p, X) \quad (6.6)$$

6.4.4 Sélectionner les meilleurs candidats

Une fois le calcul du niveau de confiance global $c(p)$ terminé, les meilleures tonies peuvent être sélectionnées. Cette sélection est simplement réalisée en conservant que les tonies ayant un niveau de confiance $c(p)$ supérieur à un seuil définit.

6.5 Pondération des tonies potentielles

Cette section présente le calcul des niveaux de confiance $c_i(p, X)$ utilisés par l'équation 6.6, c'est-à-dire :

1. Confiance sur l'intensité ;
2. Confiance sur l'octave ;
3. Confiance sur le pic.

6.5.1 Confiance sur l'intensité

Pour le violon, les notes jouées en simultanées ont généralement des intensités équivalentes. Par exemple, une tonie potentielle estimée ayant une dynamique pianissimo serait de faible confiance, si au même instant, une autre tonie potentielle estimée était trouvée avec une dynamique fortissimo. Par conséquent, il est logique d'estimer l'intensité sonore de toutes les tonies potentielles estimées et, ensuite, de calculer un niveau de confiance sur l'intensité en fonction de la tonie ayant la plus grande intensité à ce moment spécifique.

Ce niveau de confiance est calculé selon l'équation 6.7, où $loud(p, X)$ correspond à une fonction donnant l'intensité sonore d'une tonie p , où les valeurs q correspondent à toutes les autres tonies potentielles estimées pour une même trame d'échantillons audio et α est une variable ajustée pour qu'elle soit supérieure ou égale à un, cette dernière permet d'augmenter le niveau de confiance des tonies ayant une intensité sonore légèrement plus faible que la tonie ayant la plus grande intensité sonore. La meilleure fonction pour obtenir l'intensité sonore ($loud()$) et la meilleure valeur α sont définies à la sous-section 6.5.5.

$$c_1(p, X) = \min \left[\alpha \frac{loud(p, X)}{\max_q [loud(q, X)]}, 1.0 \right] \quad (6.7)$$

6.5.2 Confiance sur l'octave

Lorsque deux notes séparées par une octave sont jouées en simultanées, l'amplitude spectrale correspondant à la fréquence fondamentale de la note de tonie supérieure ainsi que l'amplitude de ses harmoniques sont superposées à celles de la note de tonie inférieure. Même si cette superposition crée une structure spectrale globale très similaire à celle d'une seule note, dans le domaine temporel, les deux formes d'ondes créées par chacune des notes s'annuleront périodiquement, produisant un battement audible. Par conséquent, en écoutant ce battement, le musicien peut déterminer que deux notes différentes d'un ou de plusieurs octaves sont jouées en simultanées.

En estimant toutes les tonies contenues dans un signal où une seule note est jouée, la méthode proposée trouvera une quantité considérable de tonies en erreur d'octave. Même si ces tonies existent pour le système, le musicien n'entendra qu'une seule tonie, car il n'y aura pas de battement dans le son. Dans ce cas, de toutes les tonies estimées, le musicien n'entendra que la tonie la plus basse. Par conséquent, il est logique de calculer un niveau de confiance qui compare la tonie ayant la plus haute valeur d'intensité sonore à toutes les autres tonies qui partagent le même nom (ex. C#) selon l'équation 6.8. Pour cette équation, $loud(p, X)$ correspond à une fonction donnant l'intensité sonore d'une tonie p et O est l'ensemble de toutes les tonies séparée par une ou plusieurs octaves par rapport à p , et ce, estimées à l'intérieur de la même trame d'échantillons audio.

$$c_2(p, X) = \frac{loud(p, X)}{\max_{p^+ \in O} [loud(p^+, X)]} \quad (6.8)$$

Pour trouver l'ensemble O de toutes les tonies potentielles séparées entre elles par une ou plusieurs octaves, la distance en cents par rapport à la tonie de référence p est d'abord calculée pour chacune des tonies potentielles estimées, et ce, en utilisant l'équation 6.1.

L'ensemble O de toutes les tonies correspond aux tonies ayant une distance à l'intérieur des plages $1150 < |d| < 1250$ et $2350 < |d| < 2450$, c'est-à-dire aux tonies dont la distance en cents est égale à environ une ou deux octaves inférieures ou supérieures à la tonie de référence p . Il est à noter que la fonction d'intensité sonore ($loud()$) utilisée sera présentée à la sous-section 6.5.5.

6.5.3 Confiance sur le pic

La structure fréquentielle d'un son de violon monophonique devrait avoir des pics situés à des multiples entiers de la fréquence fondamentale et devrait avoir des creux entre ces pics. Pour des signaux polyphoniques, certains creux peuvent être absents et certains pics de tonies peuvent être superposés avec un ou plusieurs autres pics en raison de la présence de tonies multiples.

En supposant, pour un signal de faible polyphonie, que la plupart des creux soient présents et que la plupart des pics ne soient pas superposés à d'autres pics, il est possible de définir un niveau de confiance $c_3(p, X)$ défini par les équations 6.9, 6.10 et 6.11, où δ_i représente un pic interpolé approximativement à la fréquence fondamentale ou à l'une de ses harmoniques et ρ_i représente des creux interpolés situés entre les harmoniques.

$$c_3(p, X) = \max \left[0, 1 - \frac{\sqrt[N+1]{\prod_{i=0}^N \delta_i}}{\sqrt[N+1]{\prod_{i=0}^N \rho_i}} \right] \quad (6.9)$$

$$\delta_i = \text{interp} \left[\max_{\Delta=-6,-5,\dots,5,6} (X[(i+1)k_p + \Delta]) \right] \quad (6.10)$$

$$\rho_i = \text{interp} \left[\max_{\Delta=-6,-5,\dots,5,6} (X[(i+1)k_p + \Delta - 0.5k_p]) \right] \quad (6.11)$$

Pour l'équation 6.9, les fonctions moyennes et médianes ont également été analysées, mais de meilleurs résultats ont été obtenus en utilisant la fonction de moyenne géométrique. La meilleure valeur trouvée pour N est présentée à la sous-section 6.5.5.

6.5.4 Paramètres testés

Les sous-sections suivantes présentent tous les paramètres testés afin d'obtenir un bon niveau de confiance pour chaque tonie potentielle estimée.

Fonction d'intensité sonore

Afin de calculer les deux niveaux de confiance décrits par les équations 6.7 et 6.8, la fonction d'intensité sonore ($loud(p, X)$) doit être définie. Puisqu'il n'y a pas de méthode simple pour calculer l'intensité sonore perçue [48, 57], les méthodes suivantes ont été évaluées.

1. Somme des amplitudes de la fréquence fondamentale et des premières harmoniques ;
2. Valeur médiane des amplitudes de la fréquence fondamentale et des premières harmoniques ;
3. Somme des amplitudes psychoacoustiques de la fréquence fondamentale et des premières harmoniques ;
4. Valeur médiane des amplitudes psychoacoustiques de la fréquence fondamentale et des premières harmoniques.

Dans le cas des méthodes 3 et 4, les amplitudes du spectre fréquentiel ont été pondérées en utilisant l'une des courbes isoniques de gain d'intensité de la figure 6.13. Ces courbes correspondent au gain des courbes d'intensité équivalentes décrites dans le document ISO 226 :2003 [30] où les gains sont calculés par rapport au niveau de pression acoustique le plus bas trouvé dans la courbe en phones correspondante.

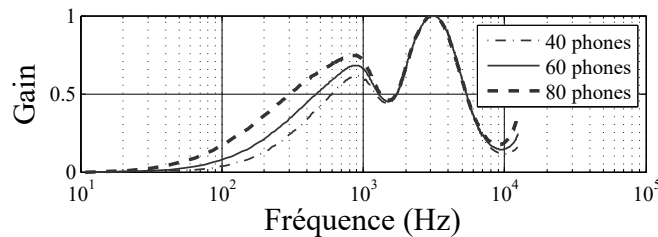


Figure 6.13 Courbes de pondération psychoacoustique

Fonction pour obtenir un niveau de confiance global

Pour chaque tonie potentielle estimée, les niveaux de confiance des équations 6.7, 6.8 et 6.9, doivent être groupés pour former un niveau de confiance global. Pour trouver la meilleure fonction de regroupement, les fonctions moyenne, moyenne géométrique, maximum, minimum et produit ont été évaluées.

Autres paramètres

Pour l'équation 6.7 différentes valeurs α ont été testées dans l'intervalle $[1.0, 2.0]$. Pour l'équation 6.9, différentes valeurs N ont été testées dans l'intervalle $[0, 7]$. Pour l'équation 6.9, seules les fonctions moyennes et médianes ont été évaluées.

6.5.5 Les meilleurs paramètres

La performance de la fonction utilisée pour calculer le niveau de confiance globale est évaluée en utilisant la distribution des tonies estimées correctement et la distribution des tonies estimées en erreur, comme illustrée à la figure 6.14, où les deux distributions sont normalisées en utilisant leur aire totale correspondante.

En référence avec la figure 6.14, le résultat de performance de la fonction correspondrait à la somme de la surface des tonies manquées et des tonies estimées en erreur (faux positifs) en fonction d'un seuil fixé. Ainsi, une bonne fonction donnerait un score qui se rapproche de zéro puisque cela impliquerait qu'il y aurait peu de tonies manquées et peu de tonies estimées en erreur. Bien que pour cet exemple fictif, le seuil ait été fixé à la valeur de 0.5, le seuil sélectionné pour l'évaluation est celui qui donne la somme la plus faible des surfaces.

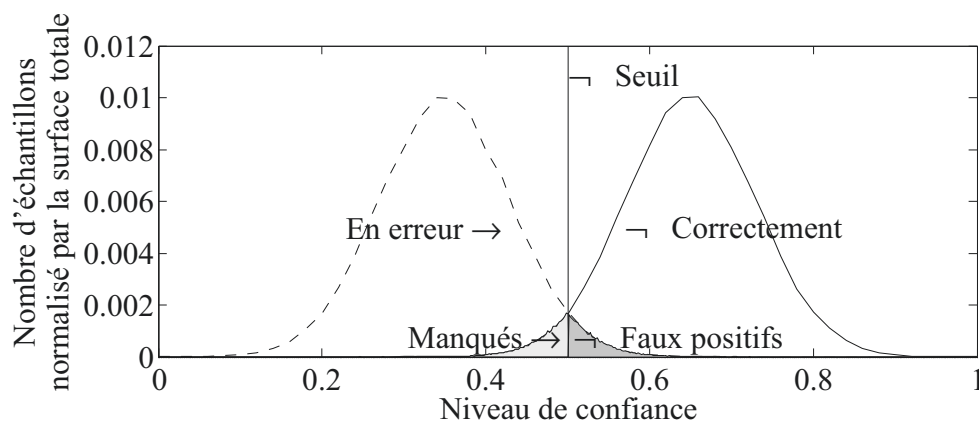


Figure 6.14 Exemple d'histogramme du niveau de confiance associé aux tonies estimées correctement et estimées en erreur

La performance a été évaluée pour chacune des combinaisons des paramètres décrits à la sous-section 6.5.4. Les paramètres donnant la meilleure performance sont listés dans le tableau 6.5.

Tableau 6.5 Meilleurs paramètres pour le calcul du niveau de confiance

Nom du paramètre	Meilleur
Fonction d'intensité sonore	Médiane psychoacoustique 6 harmoniques / 50 phones
Fonction de groupement	Moyenne
Valeur α	1.31
Valeur N	10
Fonction de groupement de l'équation 6.9	Moyenne Géométrique
Seuil	0.74

Selon ces résultats, la meilleure fonction de regroupement est définie par l'équation 6.6. Pour le calcul de l'intensité sonore, la meilleure performance a été obtenue en utilisant la méthode 4 de la sous-section 6.5.4 avec la courbe de 50 phones et en utilisant six harmoniques. Par conséquent, l'intensité sonore sera estimée en utilisant l'équation 6.12 où k_p correspond à l'indice de la raie de la TFD correspondant à la fréquence fondamentale fondamentale et $|X^*[k_p]|$ à l'amplitude de la fréquence fondamentale pondérée par la fonction psychoacoustique à la raie k_p . Le fait que la méthode 4 donne de meilleurs résultats n'est pas surprenant, puisque l'intensité perçue est liée à la réponse en fréquence irrégulière du système auditif psychophysique modélisé, ici, à l'aide de la courbe isotonique de 50 phones.

$$loud(p, X) = median(|X^*[k_p]|, |X^*[2k_p]|, \dots, |X^*[7k_p]|) \quad (6.12)$$

La figure 6.15 présente la distribution normalisée du niveau de confiance des tonies estimées avec succès et estimées en erreur en utilisant les paramètres de la table 6.5. Selon cette figure, les deux distributions sont bien séparées, car la majeure partie du niveau de confiance lié aux tonies estimées correctement a une valeur plus élevée que la majeure partie du niveau de confiance lié aux tonies estimées en erreur.

En ne gardant que les tonies potentielles ayant un niveau de confiance supérieur à une valeur définie, il est possible de réduire le nombre de tonies estimées en erreur, mais au risque de perdre certaines tonies estimées correctement. À l'aide de ce principe, il est possible de générer la figure 6.16.

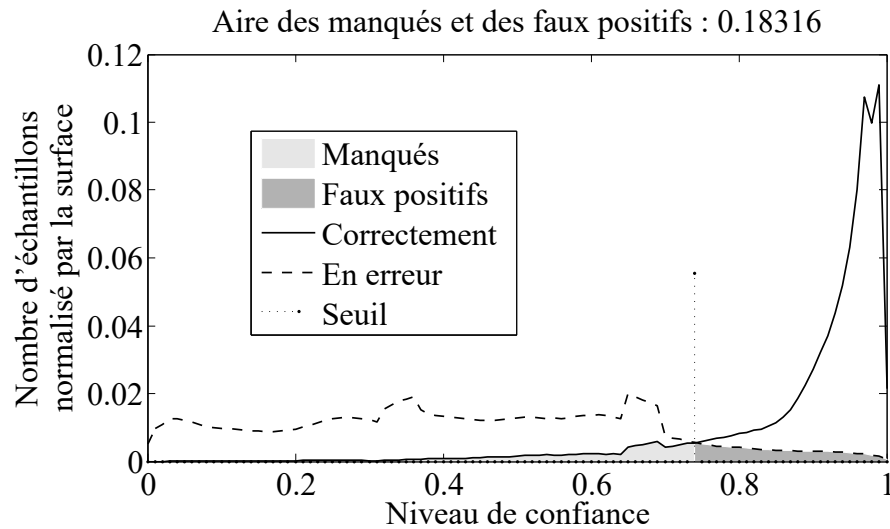


Figure 6.15 Distribution normalisée du niveau de confiance des tonies en utilisant les paramètres du tableau 6.5

En fait, la figure 6.16 permet de comparer les divers niveaux de confiance entre eux. Par exemple, en observant cette figure, il est possible de remarquer que lorsqu'un seuil donné permet d'obtenir un pourcentage de tonies correctement estimées inférieur à 92,5 %, le niveau de confiance global est plus performant, car c'est lui qui rejette le plus de tonies en erreur. En observant la courbe associée au niveau de confiance sur l'intensité, on remarque que lorsque toutes les tonies potentielles ayant un niveau de confiance inférieur à 1.0 sont rejetées, ce niveau de confiance permet d'obtenir 85 % de tonies estimées correctement et 45 % de tonies estimées en erreur.

6.6 Discussion

L'avantage de l'estimateur de tonies proposé est qu'il peut récupérer presque toutes les tonies contenues dans un signal audio de violon, et ce, en calculant seulement quelques tonies par trame comme démontré à la figure 6.11. Afin de pouvoir retirer la plupart des tonies estimées en erreur, l'estimateur de tonies proposé fournit aussi un niveau de confiance à chacune des tonies estimées. Ce niveau de confiance donne la possibilité à un système client de décider s'il désire ou non utiliser une tonie estimée. Par exemple, un système pourrait regrouper des tonies similaires et successives dans le temps pour former des notes et, ensuite, rejeter toutes les tonies estimées de faible niveau de confiance qui ne peuvent être regroupées ensemble pour former d'autres notes. Le système pourrait également supprimer toutes les notes où le niveau de confiance moyen des tonies contenues dans chacune d'entre elles est faible. Cette approche est conforme avec les approches de type (*bottom-up*, en anglais) où la décision de conserver ou de supprimer des données

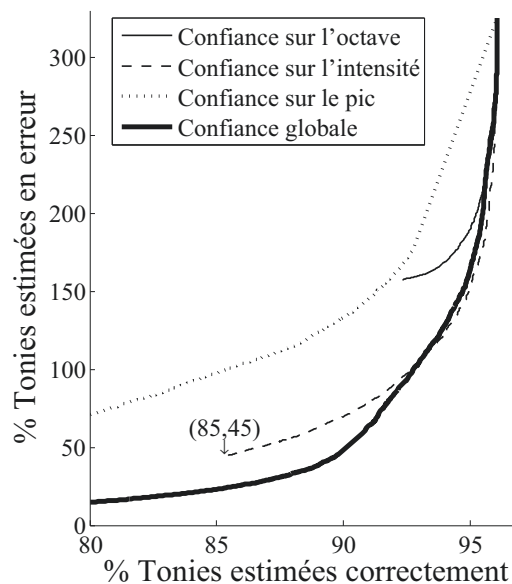


Figure 6.16 Performance pour les niveaux de confiance individuels et le niveau de confiance combiné

estimées est léguée aux modules d'abstraction de niveau supérieur, qui peuvent compter sur d'autres informations pour réaliser leurs sélections.

Il est à noter que les paramètres, tels que la longueur de trame, le pourcentage de chevauchement entre les trames et la quantité d'échantillons utilisée pour le remplissage nul, sont fixés pour l'ensemble de tous les signaux de violon analysés. Bien que cette démarche ait été préconisée pour sa simplicité de conception et de mise en œuvre, une démarche plus optimale, mais beaucoup plus laborieuse, aurait été d'analyser les effets de chacun des paramètres sur différents types de notes tels que des notes de courte et de longue durée, des notes de différentes hauteurs, des notes jouées avec différents styles (legato, staccato, pizzicato, etc.), des notes jouées à partir de violons différents et des notes jouées en monophonie et en polyphonie. Ainsi, en s'appuyant sur les résultats d'analyse, la couche de traitement du signal aurait pu être conçue pour traiter en parallèle diverses combinaisons de paramètres optimisés selon le contexte et, au final, aurait pu permettre l'estimation de nouvelles tonies à valeur de confiance élevée qui n'auraient probablement pas été estimées à partir de la version actuelle de la couche de traitement du signal.

6.7 Contributions scientifiques de la couche

La conception de la couche de traitement du signal telle que présentée dans ce chapitre a mené aux contributions scientifiques suivantes :

1. Un estimateur polyphonique de tonies fonctionnant en temps réel conçu spécialement pour le violon et efficace en présence de segment de musique de calibre professionnel.
2. Arithmétique permettant d'associer un niveau de confiance à chacune des tonies estimées.
3. Approche de type *bottom-up* où les tonies incertaines sont, malgré tout, transmises à la couche supérieure, et ce, dans le but de donner la responsabilité à la couche supérieure de conserver ou non ces tonies.

Il est à noter que les détails de performance de la couche sont présentés au chapitre 10. Le prochain chapitre décrit maintenant le fonctionnement de la couche de transcription automatique qui a pour objectif d'estimer les notes potentiellement jouées par le musicien, et ce, à partir des tonies estimées par cette dernière couche de traitement du signal.

CHAPITRE 7

COUCHE DE TRANSCRIPTION AUTOMATIQUE

Ce chapitre présente le fonctionnement de la couche de transcription automatique utilisée par le module *Analyse de la performance musicale* de la figure 5.2. La couche de transcription automatique utilise comme entrée différentes variables de configuration et les tonies potentielles estimées par la couche inférieure de traitement du signal où chacune de ces tonies est accompagnée d'une valeur de confiance ainsi que d'une intensité basée sur la psychoacoustique.

La couche de transcription automatique a pour objectif d'estimer les notes potentiellement jouées par le musicien, et ce, à partir des tonies estimées par la couche de traitement du signal. Pour ce faire, la couche rassemble principalement les tonies similaires pour former des notes potentielles telles qu'illustrées à la figure 7.2, et ce, à partir des tonies de la figure 7.1. Il est à noter que la couche de transcription automatique a, entre autres, la capacité de rejeter certaines des tonies et la capacité d'interpoler certaines tonies jugées comme manquantes.

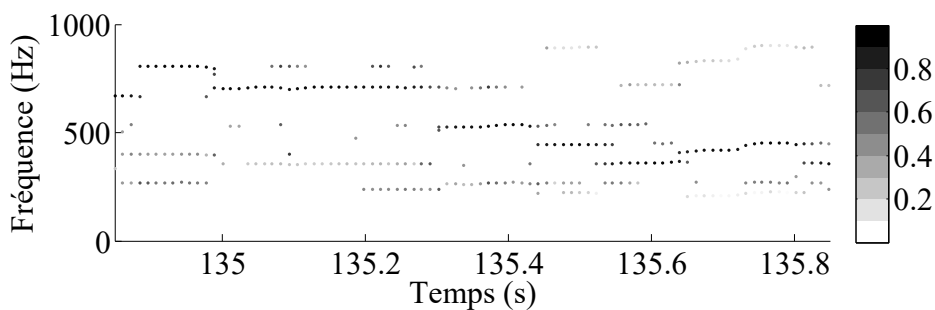


Figure 7.1 Exemple de tonies estimées par la couche de traitement du signal où l'intensité d'une tonie correspond à sa valeur de confiance

Une fois estimées, ces notes sont transmises à la couche d'alignement qui a, pour sa part, l'objectif de comparer les notes contenues dans la partition de musique à ces dernières notes potentielles, et ce, dans le but d'estimer le segment de la partition joué par le musicien et, ensuite, dans le but de le suivre en continu à travers la partition de musique. Le chapitre 8 présente avec plus de détails l'objectif et le fonctionnement de la couche d'alignement.

Tout comme pour la couche de traitement du signal, la couche de transcription automatique est basée sur une approche de type *bottom-up* puisqu'elle fournit plus de notes

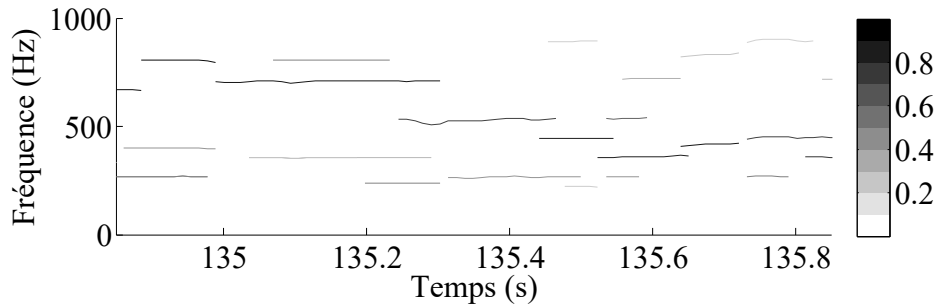


Figure 7.2 Cette figure présente un exemple de notes estimées par la couche de transcription automatique, et ce, à partir des tonies de la figure 7.1. L'intensité d'une note correspond à la moyenne des valeurs de confiance des tonies utilisées pour former la note.

que nécessaire à la couche supérieure d'alignement. En fait, la couche de transcription automatique a pour objectif d'estimer toutes les notes jouées par le musicien, et ce, malgré l'estimation de notes en erreur. Donc, plutôt que d'éliminer des notes incertaines, la couche de transcription automatique transmet ces dernières à la couche supérieure. Ainsi, la couche d'alignement est le niveau où le système proposé prend la décision d'utiliser ou non les notes qui lui ont été transmises.

En fait, la couche supérieure d'alignement n'a pas nécessairement besoin que toutes les notes estimées soient fiables à 100% puisqu'il ne faut pas perdre de vue que cette couche possède une information importante que la couche de transcription automatique ne possède pas, c'est-à-dire la partition de musique jouée par le musicien. Ainsi, comparativement à la couche de transcription automatique, la couche d'alignement peut se baser sur la partition de musique pour décider de rejeter ou non une note potentielle. Par exemple, une note estimée de faible intensité qui n'existe pas dans le segment de partition joué par le musicien pourrait être rejetée.

En comparant les notes estimées aux notes de la partition au lieu de comparer les tonies estimées aux notes de la partition, la couche d'alignement utilise moins de ressource mémoire et CPU pour estimer l'alignement optimal comme il y a moins de notes que de trames, et ce, en vue de pouvoir éventuellement réaliser un système qui permettra de répondre à la question de recherche, soit un système capable de fonctionner en temps réel de type meilleur effort et en présence d'une prestation musicale de niveau avancé, c'est-à-dire composée de notes rapides et polyphoniques.

Les détails concernant le fonctionnement de la couche de transcription automatique sont présentés dans les pages qui suivent.

7.1 Mise en contexte

La figure 7.3 illustre un exemple de tonies estimées par la couche de traitement du signal présentée au chapitre 6. Les tonies sont estimées pour chacune des trames d'échantillons audio, et ce, en se basant sur la procédure de chevauchement de trames pour augmenter la résolution temporelle telle qu'illustrée à la figure 7.4 où $x_i[n]$ représente un échantillon audio discret d'index n d'une trame i de N échantillons. Le signal audio utilisé comme entrée contient deux notes de violon consécutives jouées avec un vibrato. Dans cet exemple, plusieurs tonies par trame ont été estimées dont certaines sont visiblement estimées en erreur.

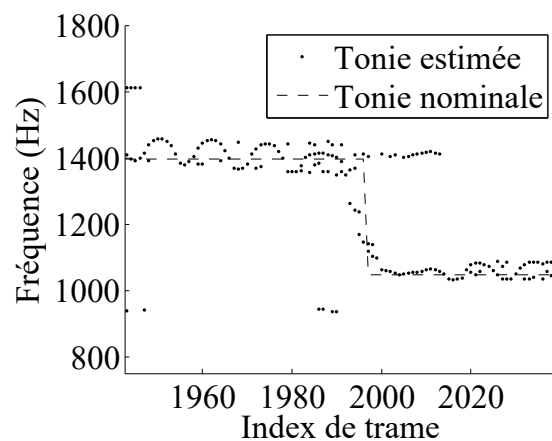


Figure 7.3 Exemple de tonies estimées

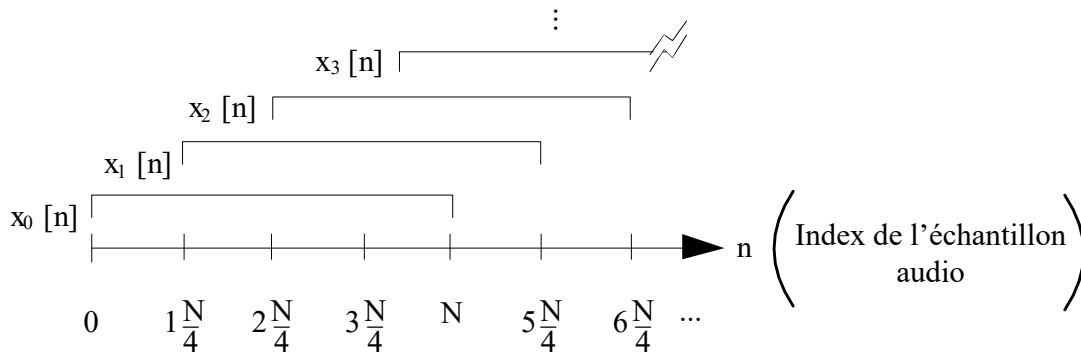


Figure 7.4 Exemple de trames chevauchées

L'approche intuitive pour lier toutes les tonies appartenant à une même note est de convertir la tonie estimée en indice MIDI (Musical Instrument Digital Interface), puis de lier ensemble toutes les tonies consécutives ayant le même indice midi. L'indice midi I est défini à l'équation 7.1 où $[x]$ est la fonction d'arrondissement, p est la tonie, p_{ref} et I_{ref} corres-

pondent à la tonie de référence et à l'index MIDI de cette tonie de référence correspondant habituellement au La440, soit un index midi de 69 et une tonie de 440Hz [60].

$$I = [12\log_2(p/p_{ref})] + I_{ref} \quad (7.1)$$

Bien que cette approche soit simple et, pour certains cas, efficace, elle présente tout de même quelques inconvénients qui peuvent diminuer considérablement la performance d'un système. Le principal problème est qu'elle suppose que les instruments de musique sont accordés en fonction d'une tonie de référence, soit globalement La440. Cependant, il n'y a aucune garantie qu'un instrument sera accordé comme tel. De plus, pour certains instruments tels que le violon, la tonie d'une note est ajustée en direct par l'interprète et, encore une fois, il n'y a aucune garantie que le musicien joue chacune des notes avec justesse. Enfin, même si un instrument est accordé près de la tonie de référence et joué par un musicien professionnel, il se peut que la tonie soit différente de la plage attendue. En fait, cela peut souvent être le cas pour des notes jouées avec un vibrato puisque le vibrato peut, dans certains cas, couvrir largement et même dépasser la plage attendue. C'est le cas de la figure 7.5 pour une note de vibrato extraite de l'enregistrement audio professionnel de violon de référence [59]. Pour illustrer le problème, l'équation 7.1 a été utilisée pour lier ensemble toutes les tonies appartenant à la note de la figure 7.5. Tel qu'illustré à la figure 7.6, même si une seule note a été jouée, des notes multiples ont été estimées par l'approche typique.

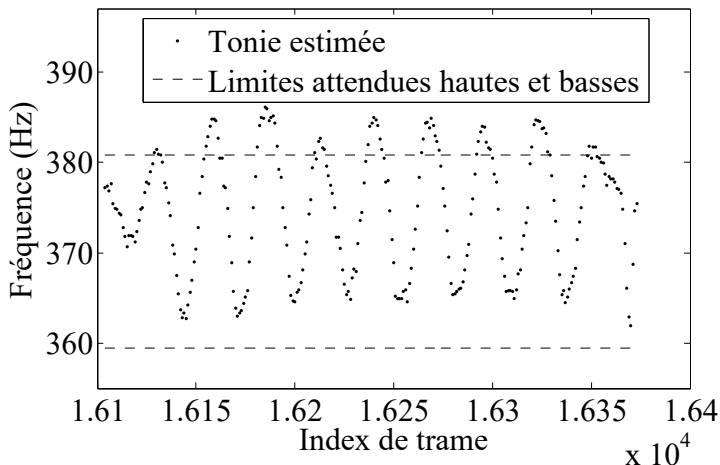


Figure 7.5 Exemple de tonies estimées à l'extérieur de la plage attendue

Sans opérations supplémentaires, la présence de tonies manquantes peut également entraîner l'estimation inadéquate de plusieurs notes lorsqu'une seule note est jouée, car les tonies

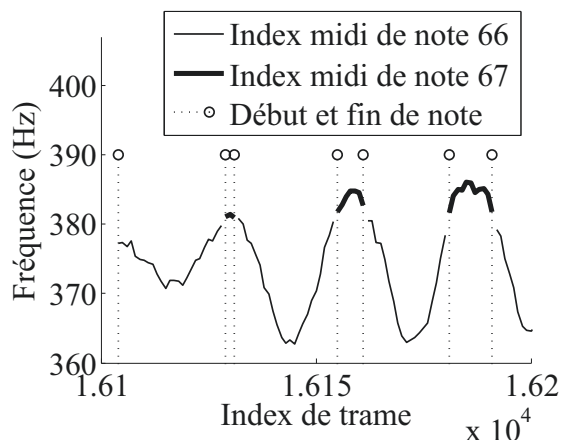


Figure 7.6 Notes résultantes en utilisant la méthode typique

manquantes seront interprétées comme des silences. La figure 7.7 illustre un exemple où certaines tonies sont manquantes. Pour ce cas, la couche d'alignement présentée au chapitre 6 a été utilisée pour estimer les tonies d'un segment audio contenant deux notes jouées en simultanées extraites de l'enregistrement audio professionnel de référence [59].

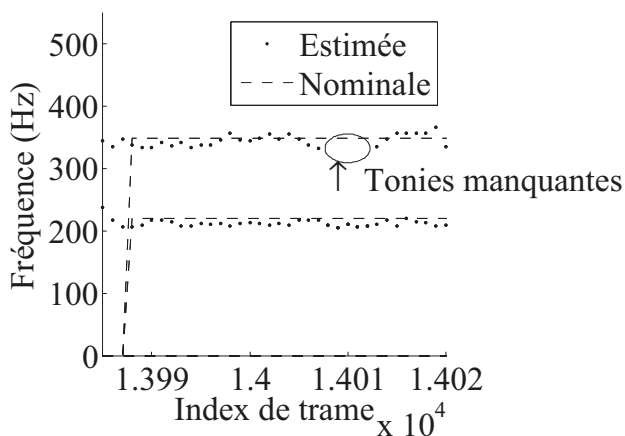


Figure 7.7 Exemple de tonies manquantes

Afin de surmonter tous ces inconvénients, la couche de transcription automatique présentée dans ce chapitre utilise plusieurs stratégies. La section suivante présente le fonctionnement de cette couche.

7.2 L'estimateur de notes proposé

Même si la procédure proposée peut être utilisée pour des applications hors ligne, elle a été développée pour des applications fonctionnant en temps réel de type meilleur effort. Par conséquent, la procédure s'exécute en continu pendant que les échantillons audio sont capturés à partir du microphone en utilisant la carte audio de l'ordinateur personnel pour

convertir le signal analogique en signal numérique. A chaque nouvelle trame d'échantillons audio, les tonies sont estimées par l'estimateur polyphonique de tonies du chapitre 6. Ensuite, les cinq étapes de la figure 7.8 sont appliquées une à une.

Au départ, la *Liste de segments* est vide. La première étape utilise une à une les tonies estimées pour créer des segments ou pour modifier des segments existants de la *Liste de segments*. Pour sa part, la dernière étape retire tous les segments contenus dans la *Liste de segments* qui représente une note à part entière. Ceci dit, un segment correspond à un vecteur de tonies analogues chronologiquement ordonnées où une seule tonie par trame est enregistré et où il n'y a pas de tonie manquante, c'est-à-dire, une tonie associée à chaque index de trame contenu entre le premier et le dernier index du segment.

Pour ainsi dire, un segment contient les tonies contenues dans un fragment continu d'une note et lorsqu'un segment contient toutes les tonies contenues dans une note, il est déplacé de la *Liste de segments* vers la *Liste de notes*, et ce, lors de la dernière étape, soit l'étape *Retirer les complets*. Il est à noter que tout au long du traitement, chacune des cinq étapes est effectuée sur chacun des segments contenus dans la *Liste de segments*.

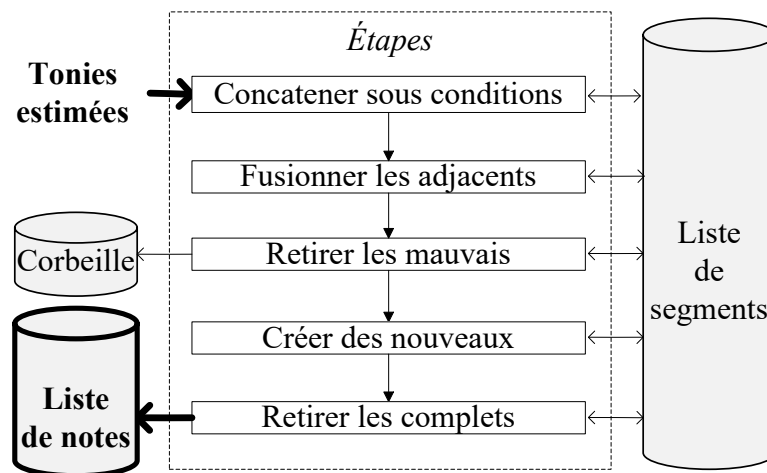


Figure 7.8 Étapes de la procédure

L'étape *Fusionner les adjacents* est utilisée pour joindre des segments jugés comme faisant partie d'une même note. Pour être fusionnés, deux segments doivent être séparés entre eux par un petit intervalle de temps.

Comme discuté précédemment, la *Liste de segments* augmente au fil du temps. Pour gérer l'expansion de cette liste, l'étape *Retirer les mauvais* est appliquée pour supprimer une multitude de segments redondants.

L'étape *Concatener sous conditions* essaie de concaténer chaque nouvelle tonie estimée à chacun des segments contenus dans la *Liste de segments*. Pour le cas où une tonie estimée ne pourrait être concaténée à aucun des segments, l'étape *Créer des nouveaux* créerait un nouveau segment d'un élément avec cette tonie et l'ajouterait dans la *Liste de segments*.

Enfin, en référence avec l'étape *Retirer les complets*, une fois qu'un segment est terminé, il est supprimé de la *Liste de segments* pour former une entité supérieure; c'est-à-dire, une note de musique. Les sous-sections suivantes présentent en détail le fonctionnement de chacune de ces étapes.

7.2.1 Étape : *concatener sous condition*

Le but de l'étape de *Concaténation conditionnelle* est de concaténer chacune des tonies entrantes avec des segments existants de la *Liste de segments*. Pour être concaténées, les conditions suivantes doivent être satisfaites :

1. La distance absolue en cents entre la moyenne des tonies contenues dans le segment et la tonie en entrée doit être inférieure à un seuil défini v_A .
2. L'index de trame associé à la tonie d'entrée doit être consécutif à l'index de trame de la dernière tonie concaténée dans le segment.
3. La distance absolue en cents entre la tonie minimale du segment et la tonie en entrée doit être inférieure à ≈ 100 cents.
4. La distance absolue en cents entre la tonie maximale du segment et la tonie en entrée doit être inférieure à ≈ 100 cents.

Notez que, pour les résultats de performance de la couche présentés chapitre 11, v_A a été fixé à 50 cents.

Utilité des deux premières conditions

La première et la seconde condition utilisées pour l'étape *Concatener sous conditions* ont d'abord été présentées dans la référence [22] du présent auteur. La première condition assure que la tonie concaténée correspond à la tendance du segment. Pour illustrer l'efficacité de ces deux conditions, la figure 7.9 illustre un exemple où la transition de fréquence entre deux notes de violon est progressive (glissando) et où les notes sont séparées par un ton et jouées avec un vibrato. Le signal audio utilisé pour cette figure a été extrait de l'enregistrement audio professionnel de violon de référence [59].

Pour cette figure, le segment a été initialement défini comme un vecteur d'un élément qui correspond à la première tonie tracée. Toutes les tonies estimées qui ont été concaténées à ce segment en utilisant les conditions 1 et 2 ont été encerclées. Il est intéressant de

comparer la distance en fréquence entre les tonies estimées et la ligne d'intégration reflétant la progression de la moyenne du segment. Lors de la première note, cette distance est relativement petite, mais au fur et à mesure que le glissando progresse, cette distance commence à s'accroître et, éventuellement, la première condition devient fausse mettant fin à la concaténation des nouvelles tonies estimées. Afin d'illustrer ceci, la ligne pleine d'intégration de la figure 7.9 a été modifiée en ligne pointillée, et ce, à partir du moment où la première condition est devenue fausse.

Afin d'assurer la fiabilité de ces deux premières conditions, la durée de la note avant le glissando doit être suffisamment longue sinon un glissando lent pourrait rapidement augmenter la valeur moyenne des tonies provoquant l'échec de la première condition et, par conséquent, provoquant une mauvaise concaténation au segment en construction. Les deux prochaines conditions servent à corriger ce problème.

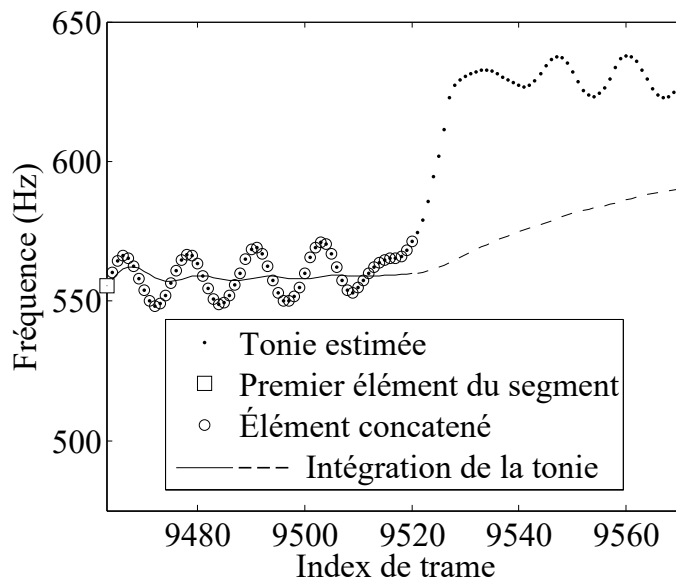


Figure 7.9 Exemple de segment généré en utilisant les conditions 1 et 2 de l'étape de *Concaténation conditionnelle* avec $v_A = 50$ cents

Utilité de la troisième et quatrième condition

Pour la majorité des cas, les deux premières conditions suffisent pour arrêter la concaténation des tonies lors de l'arrivée d'une nouvelle note. Par contre, les conditions 3 et 4 sont nécessaires pour corriger le problème mentionné précédemment.

Les conditions 3 et 4 sont utilisées pour mettre fin à la concaténation de tonies lorsque deux notes successives sont séparées par une faible distance en cents, soit une distance se rapprochant d'un demi-ton. Il est à noter que les douze notes de musique (Do, Do#, Ré, Ré#, Mi, etc.) sont séparées entre elles par un demi-ton, soit environ 100 cents [4].

Cas où plusieurs tonies respectent les conditions

Pour un segment donné, plusieurs tonies estimées pour une même trame audio pourraient respecter les quatre conditions. Ceci serait d'ailleurs le cas pour les tonies de la figure 7.3. Puisqu'une seule tonie par trame peut être concaténée à un segment, la meilleure tonie doit être sélectionnée parmi toutes les tonies potentielles.

Afin de sélectionner la meilleure tonie à utiliser pour la concaténation, la valeur de tonie attendue doit d'abord être extrapolée. Ainsi, la tonie le plus proche par rapport à cette valeur extrapolée est sélectionnée. La fonction d'extrapolation est sélectionnée en fonction du nombre de tonies contenues dans le segment (N), les extrapolations suivantes peuvent être utilisées :

- $N \geq 3$: Extrapolation parabolique de second ordre, soit en résolvant $y = ax^2 + bx + c$ à partir des trois dernières tonies concaténées.
- $N = 2$: Extrapolation linéaire, soit en résolvant $y = bx + c$ avec les deux dernières tonies concaténées.
- Autrement : Sélection de la tonie la plus proche de la tonie contenue dans le segment.

Cas où une tonie pourrait être concaténée à plusieurs segments

Pour le cas où une tonie estimée en entrée pourrait être concaténée à plusieurs segments, la tonie est simplement concaténée à chacun des segments. Cela permet d'éviter de sélectionner le mauvais segment pour la concaténation, cependant, cela aura pour conséquence de créer des segments redondants, c'est-à-dire des segments qui sont différents les uns des autres par quelques tonies seulement. Notez que les segments redondants seront supprimés lors de l'étape *Retirer les mauvais*.

7.2.2 Étape : *Fusionner les adjacents*

En utilisant uniquement l'étape *Créer des nouveaux* définie à la section 7.2.4 et l'étape *Concatener sous conditions* définie à la section 7.2.1, plusieurs segments consécutifs appartiennent souvent à une seule et même note jouée. Ce problème de segments multiples par note peut provenir de différentes sources, comme en présence de tonies manquantes telles qu'illustrées à la figure 7.10. Pour résoudre ce problème, l'étape *Fusionner les adjacents* est utilisée pour lier ensemble les segments suspectés de faire partie d'une même note. Les deux sous-sections suivantes présentent deux cas courants pouvant créer plusieurs segments pour une seule et même note.

En présence de tonies manquantes

Tel que discuté précédemment et démontré par la figure 7.10, plusieurs segments peuvent être créés lorsqu'une seule note est jouée. Cela peut être dû à la deuxième condition

de l'étape *Concatener sous conditions* qui spécifie que l'indice de trame de la tonie à concaténer doit être consécutif à celui de la dernière tonie concaténée dans le segment.

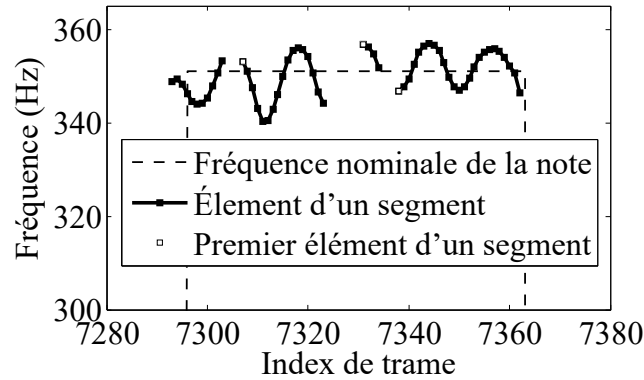


Figure 7.10 Exemple de segments multiples par note en présence de tonies manquantes

Voici quelques exemples de cas pouvant causer des tonies manquantes :

1. Erreur d'estimation de tonie telle qu'une erreur d'octave sur la tonie;
2. Incident involontaire du musicien;
3. Tonie de faible intensité en présence de notes jouées faiblement (ex. pianissimo);
4. Problème technique de l'instrument de musique.

En présence d'un vibrato

En raison de la première condition décrite à la section 7.2.1, la concaténation de tonies peut se terminer avant son temps en présence d'un vibrato marqué. Dans un tel cas, la note sera décomposée en plusieurs segments continus tels qu'illustrés à la figure 7.11, et ce, en utilisant le même seuil de concaténation (v_A) que pour la figure 7.9.

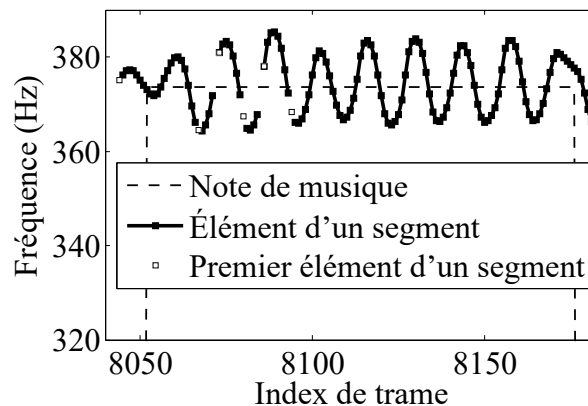


Figure 7.11 Exemple de segments multiples en présence d'un vibrato

L'augmentation de la résolution temporelle et l'augmentation du pourcentage de chevauchement de trames peut réduire ce problème, mais cela peut également entraîner une charge importante sur le processeur. La prochaine étape présente la solution proposée pour résoudre la majorité de ces problèmes, et ce, sans recourir à une charge excessive sur le processeur.

Fonctionnement de l'étape *Fusionner les adjacents*

L'étape *Fusionner les adjacents* est assez simple. Elle consiste, à fusionner des segments rapprochés dans le temps qui sont soupçonnés de partager la même tonie moyenne, c'est-à-dire, soupçonnés de faire partie d'une même note. La fusion est effectuée selon les étapes suivantes :

1. Si nécessaire, extrapoler les tonies manquantes entre les segments.
2. Concaténer les tonies extrapolées au segment de gauche.
3. Concaténer le segment de droit au segment résultant de l'étape précédente.

Pour fusionner deux segments successifs, les conditions suivantes doivent tout de même être satisfaites :

1. L'intervalle de temps entre les deux segments doit être inférieur à une valeur définie (v_B).
2. Selon l'axe des temps, le segment de droite doit se terminer après la fin du segment de gauche.
3. La plage chevauchée en fréquence divisée par la plage de fréquences couverte par le segment gauche doit donner un résultat supérieur à une valeur définie (v_C).

Notez que, pour les résultats de performance de la couche présentés chapitre 11, les valeurs v_B et v_C ont été ajustées à 100 millisecondes et à 0.25 respectivement.

Il existe plusieurs façons d'extrapoler les tonies manquantes, il est suggéré d'utiliser l'interpolation cubique à quatre points, c'est-à-dire d'utiliser les deux dernières tonies du segment gauche et les deux premières tonies du segment droit pour calculer les coefficients a_i de l'équation polynomiale cubique $x^3 + a_1x^2 + a_2x + a_3$. La figure 7.12 présente un exemple où l'interpolation cubique à quatre points a été utilisée pour extrapoler les tonies manquantes de la figure 7.10.

Plusieurs notes consécutives de la même tonie

En présence de plusieurs notes consécutives ayant la même tonie, l'étape *Fusionner les adjacents* est sujette aux erreurs, car, de toute évidence, elle peut fusionner des segments

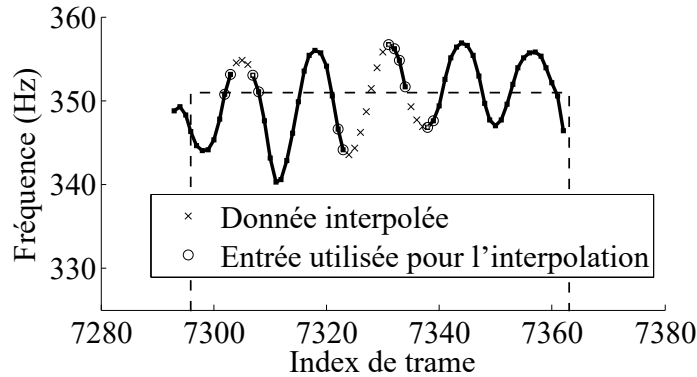


Figure 7.12 Exemple de tonies interpolées avec l'interpolation cubique à quatre points

n'appartenant pas à la même note. Par conséquent, pour la première condition répertoriée dans la sous-section 7.2.2, il est recommandé d'utiliser le plus faible intervalle de temps v_B possible entre les segments pour permettre la fusion.

En addition, il est recommandé également d'insérer toutes les notes de musique possibles dans la *Liste de notes* afin de laisser le module client décider s'il désire utiliser ou non la note de musique. Cette stratégie sera discutée plus en détail dans la section 7.4.

7.2.3 Retirer les mauvais

Sans aucune maintenance, la *Liste de segments* peut augmenter de manière significative et cela est encore plus vrai lorsque plusieurs tonies par trame sont estimées. Par conséquent, l'étape *Retirer les mauvais* peut être utilisée pour gérer la taille de la *Liste de segments*. Cette étape consiste à comparer chaque segment à chacun des segments existants afin de détecter les segments redondants.

Pour être considéré comme redondant, un segment par rapport à un autre segment doit respecter toutes les conditions suivantes :

1. Le segment commence au même moment ou après le segment de comparaison.
2. Le segment se termine au même moment ou avant le segment de comparaison.
3. La distance en cent entre la tonie moyenne des deux segments est inférieure à une valeur définie, soit v_D .
4. Selon l'axe des fréquences, les segments comparés se chevauchent à plus qu'une valeur définie, soit v_E .

Notez que, pour les résultats de performance de la couche présentés chapitre 11, les valeurs v_D et v_E ont été ajustées à 100 cents et 0.5 respectivement.

7.2.4 Étape : *Créer des nouveaux*

L'étape *Créer des nouveaux* est une étape simple qui crée de nouveaux segments d'un élément en utilisant les tonies estimées qui ne peuvent pas être concaténées à segment existant. Ces nouveaux segments sont ajoutés à la *Liste de segments*.

7.2.5 Retirer les complets

Selon l'étape *Fusionner les adjacents*, pour fusionner deux segments similaires, l'intervalle de temps entre les segments doit être inférieur à un seuil défini v_B . Par conséquent, une fois que la distance temporelle entre la dernière tonie d'un segment et la dernière tonie estimée en entrée est supérieure à cette valeur de seuil, le segment ne peut plus être mis à jour, car la concaténation et la fusion ne peuvent plus se produire pour ce segment. Ce n'est qu'alors que le segment est supprimé de la *Liste de segments* et déplacé, sous une condition, vers la *Liste de notes*.

Pour être déplacé dans la *Liste de notes*, le segment doit respecter une condition simple, la durée du segment doit être suffisamment longue pour être considérée comme une note potentielle. En pratique, cette durée devrait être un paramètre configurable défini en fonction des besoins de l'application. Par exemple, une application de suiveur de partition peut utiliser le contenu de partition musicale pour évaluer la durée la plus faible d'une note attendue.

7.3 Rejet d'une note selon son intensité sonore

Cela peut sembler une bonne idée d'enlever toutes les tonies estimées ayant une énergie sous un seuil défini. Cependant, imaginons d'abord une longue note pianissimo, soit une note de faible intensité sonore, jouée en utilisant un instrument acoustique tel qu'une flûte. Même si le musicien fait de son mieux, le volume oscillera probablement autour d'une intensité sonore nominale. Par conséquent, il n'y a aucune garantie que l'intensité sonore de chacune des tonies de la note demeurera au dessus du seuil ajusté. En fait, certaines tonies de la note pourraient dépasser le seuil et d'autres non, et ce, tout au long du jeu de la note. Dans ce cas, supprimer les tonies estimées ayant une énergie inférieure à un seuil défini peut produire plusieurs trous importants dans la note qui ne pourront pas être corrigés par la procédure proposée, soit, plus spécifiquement, par l'étape *Fusionner les adjacents* définie à la section 7.2.2.

Pour éviter que cela ne se produise, il est suggéré que la couche de traitement du signal transmette toutes les tonies estimées comme probables à la couche de transcription automatique et qu'elle laisse à cette dernière la décision de supprimer les notes, soit en

supprimant les notes ayant une intensité moyenne sous un seuil défini. De cette façon, la corrélation entre les tonies est prise en considération dans le processus de rejet.

Puisque la perception humaine de l'intensité sonore n'est pas constante à travers l'échelle des fréquences. L'intensité sonore doit être estimée selon la psychoacoustique. Afin de s'approcher de l'intensité perçue d'une tonie, il est proposé d'utiliser la méthode des courbes d'intensité isotoniques décrites à la sous-section 6.5.4 du chapitre 6. D'ailleurs, le travail revient à la couche de traitement du signal de transmettre cette intensité perceptuelle comme elle réalise déjà ce calcul pour calculer la valeur de confiance sur l'intensité décrite à la section 6.5.1.

À titre indicatif, l'intensité d'une tonie p peut être calculée selon l'équation 7.2 où X est la trame Transformée de Fourier Discrète (TFD), k_p est la fréquence d'indice p correspondant à la fréquence fondamentale de la tonie et $\Psi[k]$ est la version discrète des courbes de gain isotonique de la figure 7.13 correspondant à l'échelle de fréquence de X . Puisque ces trois courbes de gain ont des formes similaires, la courbe de gain du moyenne de 60 phones est utilisée pour les calculs, et ce, en tant qu'approximation aux autres courbes.

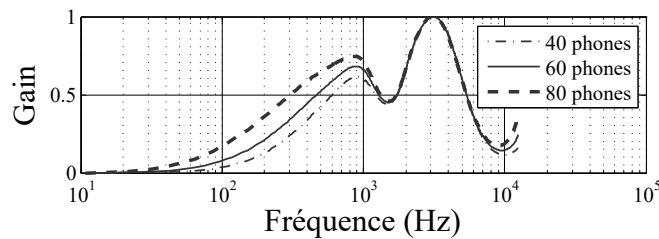


Figure 7.13 Courbes de gain isotonique

$$\text{loud}(p, X) = \text{median} \left(\begin{array}{c} |X[k_p]| \times \Psi[k_p] \\ |X[2k_p]| \times \Psi[2k_p] \\ \dots \\ |X[Nk_p]| \times \Psi[Nk_p] \end{array} \right) \quad (7.2)$$

Afin d'obtenir une meilleure approximation, au lieu d'utiliser une valeur entière d'indice p pour k_p , sa valeur réelle a été utilisée. Cependant, les valeurs $[nk_p]$ ont pour leur part été arrondies pour pouvoir être multipliées par la valeur de $\Psi[k]$ discrète respective.

7.4 Attribution d'un niveau de confiance aux notes estimées

Un estimateur polyphonique de tonies conventionnel traitera chacune des trames sans considérer les trames précédentes. Cependant, même si cet aveuglement simplifie grandement la conception de l'estimateur de tonies, dans les faits, il y a une corrélation importante entre les trames.

Cela dit, une fois toutes les tonies regroupées pour former une note de musique, un nouveau niveau d'abstraction est créé et, par le fait même, de nouvelles observations peuvent être réalisées en se basant maintenant sur des éléments de la théorie musicale. Par conséquent, un niveau de confiance peut être calculé en se basant sur cette dernière théorie pour chacune des nouvelles notes estimées, et ce, afin d'aider un module client, tel qu'un système d'accompagnement automatique, à décider s'il désire ou non utiliser une note spécifique pour réaliser sa tâche.

Fournir un niveau de confiance à chacune des nouvelles notes estimées est, en un sens, une meilleure stratégie que de supprimer les notes suspectées d'être estimées en erreur puisque ces notes sont toujours disponibles et la décision de les utiliser est donnée à des modules d'abstraction plus élevés qui peuvent s'appuyer sur d'autres observations pour réaliser leurs tâches, par exemple un tourneur automatique de pages qui pourrait s'appuyer sur les notes contenues dans la partition de musique pour décider s'il désire ou non utiliser une note en particulier. Les sous-sections suivantes présentent différents niveaux de confiance de notes utilisés par le système proposé.

7.4.1 Niveau de confiance pour le début et la fin d'une note

La perception de la musique d'un musicien dépend de nombreux facteurs, tels que son expérience musicale personnelle et sa physiologie héritée. Même avec la technologie d'aujourd'hui, cette complexité n'est toujours pas reproductible par programmation, et ce, en utilisant les algorithmes de traitement du signal et les notions psychoacoustiques actuels. Par conséquent, les estimateurs de tonies actuels estimeront des tonies qui seront effectivement perçues et d'autres qui seront considérées comme sans importance par un musicien.

Cela peut d'ailleurs être le cas en présence d'écho causé par la pièce ou en présence de réverbération de l'instrument. La réverbération après le relâchement d'une note, par exemple, peut se produire lorsqu'un violoniste arrête de jouer une note sur une corde, bien que la note soit terminée dans son esprit, si la corde reste intacte, la note risque de demeurer perceptible. Bien entendu, l'amplitude sonore sera probablement très faible par

rapport à l'amplitude sonore juste avant le relâchement, mais elle peut être suffisamment élevée pour être détectée par un estimateur de tonies.

Pour illustrer ce type d'effet sur l'estimation des tonies, la figure 7.14 présente un exemple pour deux notes de violon consécutives jouées sur la même corde. Pour cet exemple, des tonies sont encore estimées malgré le relâchement des notes. La figure 7.15 présente l'évolution de l'intensité sonore de la deuxième note de la figure 7.14 ($\approx 930\text{Hz}$). À partir de cette figure, on peut observer que, effectivement, l'intensité sonore est très faible suite au relâchement de la note. Comme la réverbération de l'instrument et l'écho de la pièce sont normalement présents tout au long de la pièce de musique, cet exemple est donc un parmi tant d'autres retrouvés dans l'enregistrement professionnel de violon de la référence [59].

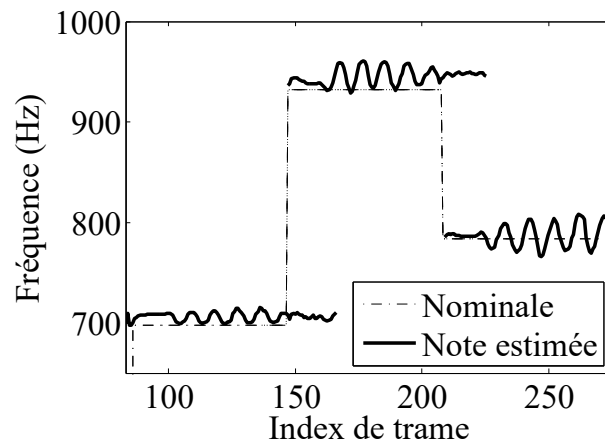


Figure 7.14 Exemple de tonies estimées après le relâchement de la note dues l'écho de la pièce

Des tonies peuvent également être estimées même si aucune note n'a été officiellement jouée par le musicien. Par exemple, anticipant une note future, un guitariste peut mettre le doigt sur la touche avant même de pincer la corde. Le coup de doigt sur la touche peut parfois produire suffisamment d'énergie pour faire résonner la note. Évidemment, l'intensité sonore de la note peut ne pas être aussi intense que celle d'une note pincée, mais elle peut être suffisamment forte pour donner une estimation précoce du début de la note.

Par conséquent, pour les raisons énumérées ci-dessus, toutes les techniques d'estimation de notes sont sujettes à des erreurs sur le moment de début et de fin des notes estimées. Afin de diminuer les impacts sur la performance du système proposé, un niveau de confiance pour le début et la fin de chacune des notes est calculé tel que décrit dans les sous-sections suivantes.

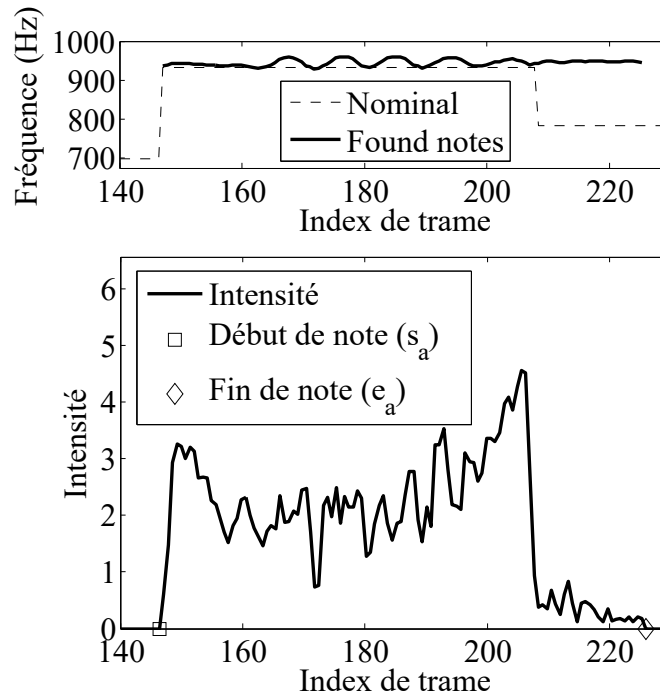


Figure 7.15 Intensité de la seconde note estimée de la figure 7.14 avec la mauvaise fin estimée résultant de l'écho

Rechercher des positions de début et de fin potentielles

La première étape consiste à analyser la courbe d'intensité sonore de chacune des notes estimées afin de trouver d'autres positions possibles de début et de fin. Les positions de fin possibles peuvent être trouvées en utilisant l'organigramme illustré à la figure 7.16, où $L[n]$ est l'intensité de la note à l'index de trame n et v_F est un paramètre configurable dans l'intervalle $]0, 1[$. Les positions de départ possibles peuvent également être trouvées en utilisant l'organigramme de la figure 7.16, mais en traitant les intensités sonores à l'envers, soit en commençant du dernier index, et ce, jusqu'au premier index ($n = 0$).

Afin d'obtenir de bons résultats, il est suggéré de lisser la courbe d'intensité avec filtre médian. Notez également que, pour les résultats de performance de la couche présentés chapitre 11, la valeur de v_F a été ajustée à 0.5.

Décomposer en plusieurs notes

Cette étape est effectuée pour chacun des débuts et fins de notes possibles. Elle consiste à ajouter chacune des notes possibles à la *Liste de notes*. La figure 7.17 illustre un exemple fictif pour expliciter davantage cette étape. En référence à la légende de la figure, pour la note estimée, deux débuts et trois fins possibles ont été identifiés en utilisant la procédure

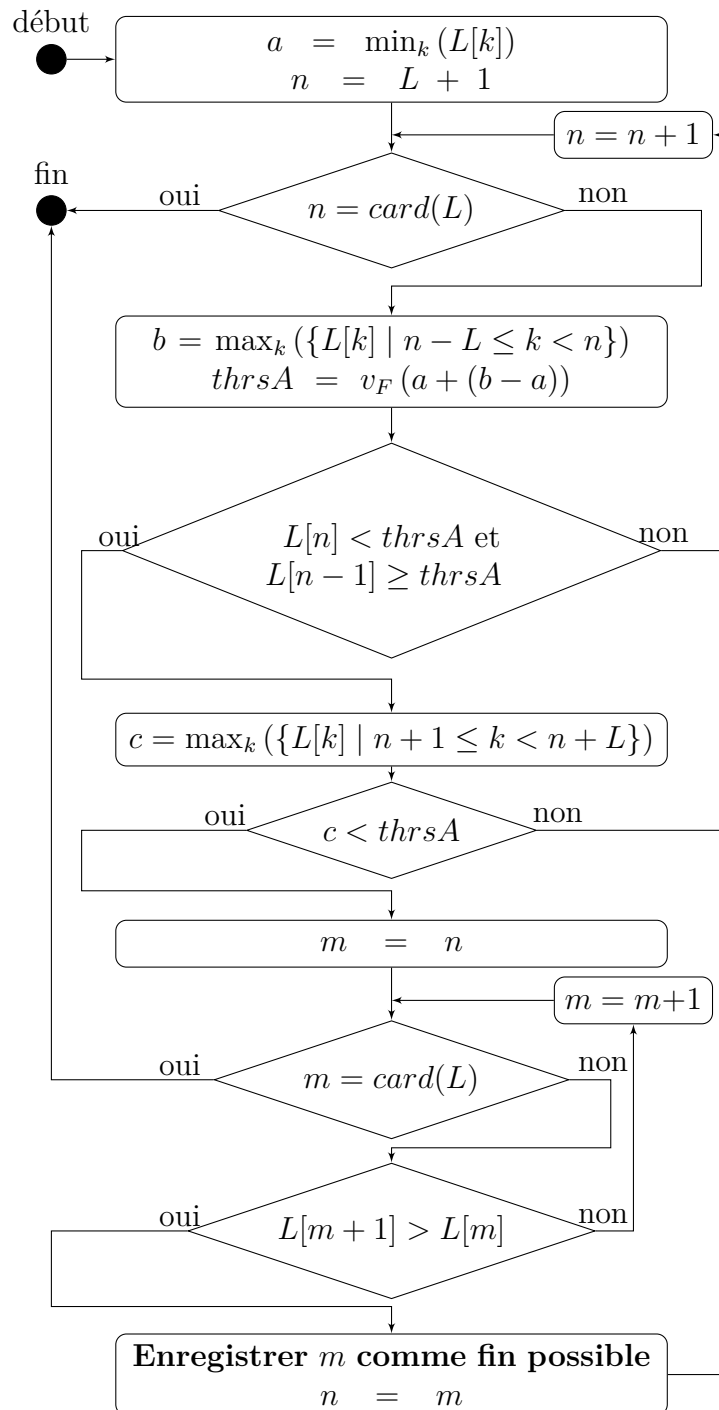


Figure 7.16 Procédure pour trouver toutes les fins possibles de note

de la figure 7.16. Comme le montre la figure 7.18, en utilisant toutes les positions de début et de fin possibles, la note de la figure 7.17 peut prendre cinq autres formes.

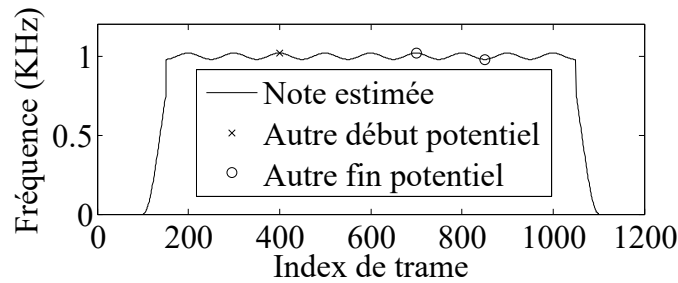


Figure 7.17 Exemple de décomposition en plusieurs notes

Dans le but de réduire le nombre de formes potentielles, certaines peuvent être considérées comme improbables en comparant d'abord chacun des points de leur courbe d'intensité sonore $L[k]$ à un seuil relatif thr selon l'équation 7.3 et en supprimant chacune des formes de note où l'intensité sonore descend sous le seuil thr pour plus de v_H millisecondes, et ce, avant de revenir au-dessus de ce seuil. Ce rejet est basé sur l'hypothèse que l'intensité d'une note restera, constante, ou augmentera (crescendo), ou diminuera (diminuendo), mais ne fera jamais une diminution significative suivie d'une augmentation significative, ce qui est généralement le cas pour la très grande majorité des notes. Notez que ce rejet réduit la quantité totale de notes en surplus d'environ 5%. Notez également que, pour les résultats de performance de la couche présentés chapitre 11, nous avons défini v_G et v_H respectivement à 0,15 et 75 millisecondes.

$$thr = \min_k(L[k]) + v_G (\max_k(L[k]) - \min_k(L[k])) \quad (7.3)$$

Enfin, une fois les formes improbables supprimées, chacune des formes restantes sera ajoutée à la *Liste de notes* avec deux niveaux de confiance pour le début et la fin respectivement qui seront définis à la section suivante.

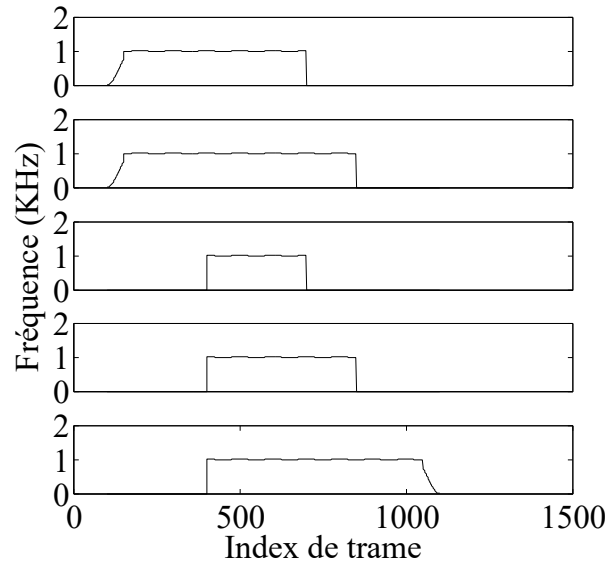


Figure 7.18 Notes découlants de la note illustrée à la figure 7.17

Niveaux de confiance de début et de fin

À cette étape, deux niveaux de confiance pour le début et pour la fin compris entre $[0, 1]$ sont calculés pour chaque note contenue dans la *Liste des notes* afin d'aider un module client à décider quelle note utiliser lorsque plusieurs notes d'un même index midi se chevauchent dans le temps, mais avec des moments de début et/ou de fin différents. Au début, chacune des notes x contenues dans la *Liste des notes* est initialisée avec un niveau de confiance de début et de fin c_x égal à un. Ensuite, chacune des notes a est comparée, une par une, à chacune des notes qui la chevauchent dans le temps b possédant le même index midi que a . Cette comparaison est réalisée en utilisant l'équation 7.4 où c_a remplacera le niveau de confiance de début ou de fin existant de la note a , et ce, si sa valeur est inférieure à celle existante. Ce processus est répété pour chacune des notes contenues dans la *Liste de notes*. Notez que, puisque le niveau de confiance de début et de fin a été initialisé à un, une note qui n'a pas de note en chevauchement et d'index commun obtiendra automatiquement un niveau de confiance de début et de fin égal à un.

$$c_a = \begin{cases} \min\left(1, \frac{\mu_{a \setminus b}}{\mu_{a \cap b}}\right) & \text{si } \mu_{a \setminus b} \text{ n'est pas vide} \\ 1 - \frac{\mu_{b \setminus a}}{\mu_{a \cap b}} & \text{si } \mu_{b \setminus a} < \mu_{a \cap b} \\ \frac{\mu_{a \cap b}}{\mu_{b \setminus a}} & \text{autrement} \end{cases} \quad (7.4)$$

Avant le calcul de l'équation 7.4, les équations 7.5 à 7.9 doivent être calculées. Les équations 7.5 et 7.6 sont utilisées pour trouver l'index de début s et de fin e de la partie

chevauchée dans le temps des deux notes où s_x et e_x correspondent au moment de début et au moment de fin d'une note x tel qu'indiqué dans l'exemple figure 7.15.

$$s = \max(s_a, s_b) \quad (7.5)$$

$$e = \min(e_a, e_b) \quad (7.6)$$

L'équation 7.7 calcule la moyenne $\mu_{a \cap b}$ de toutes les valeurs d'intensité sonore $L[n]$ contenues dans la partie superposée temporellement des deux notes, soit a et b . Notez que pour cette partie de recouvrement $L_a[n]$ et $L_b[n]$ sont considérés comme ayant les mêmes valeurs puisque les notes qui se chevauchent sont le résultat d'une seule note segmentée en plusieurs notes comme discuté précédemment dans la sous-section 7.4.1.

$$\mu_{a \cap b} = E \{L_a[n] | s \leq n \leq e\} \quad (7.7)$$

Les équations 7.8 et 7.9 correspondent à la moyenne de toutes les valeurs d'intensité sonore qui ne se chevauchent pas pour les notes a et b respectivement, soit $\mu_{a \setminus b}$ et $\mu_{b \setminus a}$.

$$\mu_{a \setminus b} = E \{L_a[n] | (s_a \leq n < s) \cup (e < n \leq e_a)\} \quad (7.8)$$

$$\mu_{b \setminus a} = E \{L_b[n] | (s_b \leq n < s) \cup (e < n \leq e_b)\} \quad (7.9)$$

Enfin, en référence avec l'équation 7.4, le niveau de confiance de début et de fin correspond simplement à un rapport entre la valeur d'intensité sonore moyenne de la partie non chevauchée et de la partie chevauchée selon que seule une partie de la note a est chevauchée ou selon que la note a est complètement chevauchée par la note b .

7.4.2 Niveau de confiance des notes selon l'octave

Un estimateur de tonies est sujet aux erreurs d'octave. Par conséquent, pour une trame donnée, même en présence de deux tonies différentes entre elles par une octave ou plus et possédant des valeurs d'intensité sonore équivalente, il est suggéré de toutes les transmettre à l'entrée de procédure décrite à la figure 7.8. Ainsi, certaines tonies seront simplement supprimées, car elles ne seront pas transformées en notes de musique. Pour les notes en trop résultantes des tonies estimées en erreur d'octave, nous suggérons de fournir à chaque

nouvelle note un niveau de confiance lié à l'octave qui sera ensuite utilisé par le module client pour décider s'il désire ou non utiliser la note pour accomplir sa tâche.

Pour calculer le niveau de confiance d'une note selon l'octave, il est proposé de calculer l'intensité sonore correspondant aux tonies estimées selon la méthode décrite dans la section 7.3. Ensuite, toutes les tonies différentes d'une ou de plusieurs octaves sont comparées à la tonie de ce groupe qui a l'intensité sonore la plus élevée afin d'obtenir un niveau de confiance selon l'octave. Cela peut être fait en utilisant l'équation 7.10 où $c_p[n]$ est le niveau de confiance d'octave d'une tonie p estimée à l'index de trame n , $L_p[n]$ est l'intensité sonore de la tonie p et O_p est l'ensemble de toutes les tonies q différentes d'une ou de plusieurs octaves par rapport à p à l'index de trame n . Notez que la tonie p est également incluse dans l'ensemble O_p .

$$c_p[n] = \frac{L_p[n]}{\max_{q \in O_p} (L_q[n])} \quad (7.10)$$

Enfin, le niveau de confiance c_n d'une note n en fonction de l'octave est simplement la moyenne de tous les N niveaux de confiance d'octave $c_p[n]$ de tonies faisant partie de cette note selon l'équation 7.11.

$$c_n = \frac{1}{N} \sum_n^N c_p[n] \quad (7.11)$$

7.4.3 Niveau de confiance d'une note selon les tonies la constituant

La couche de traitement du signal telle que décrite au chapitre 6 transmet un niveau de confiance à chacune des tonies estimées. Par conséquent, un niveau de confiance de note en fonction de ses niveaux de confiance de tonie peut être obtenu en calculant simplement la valeur moyenne de tous les niveaux de confiance de tonies contenus dans la note. Encore une fois, la valeur moyenne donne un aperçu global qui aide le module client dans sa prise de décision.

7.5 Discussion

En bref, la couche de transcription automatique propose plusieurs stratégies pour réduire les effets négatifs causés par les tonies estimées en trop ou manquantes. De plus, la couche propose aussi différents niveaux de confiance associés à chacune des notes estimées pour aider un module client à identifier la plupart des notes estimées en erreur, telles que les notes en erreur d'octave et les notes avec une position de début ou de fin incorrecte.

7.6 Contributions scientifiques de la couche

La conception de la couche de transcription automatique telle que présentée dans ce chapitre a menée aux contributions scientifiques suivantes :

1. Une procédure fonctionnant en temps réel et supérieure à la procédure typique (voir la section 10.3 du chapitre 11).
2. Différentes stratégies pour estimer adéquatement les notes de musique, et ce, malgré la présence de tonies manquantes ou estimées en extra.
3. Arithmétique permettant d'associer un niveau de confiance à chacune des notes estimées.
4. Approche de type *bottom-up* où les notes incertaines sont, malgré tout, transmises à la couche supérieure, et ce, dans le but de donner la responsabilité à la couche supérieure de conserver ou non ces notes.

Il est à noter que la performance de la couche de transcription automatique est présentée au chapitre 11. Le prochain chapitre décrit maintenant le fonctionnement de la couche d'alignement qui a pour objectif de suivre le musicien à travers la partition de musique, et ce, à partir des notes estimées par cette dernière couche.

CHAPITRE 8

COUCHE D'ALIGNEMENT

Ce chapitre présente le fonctionnement de la couche d'alignement utilisée par le module *Analyse de la performance musicale* de la figure 5.2. La couche d'alignement utilise comme entrée les variables de configuration, les notes potentielles détectées par la couche de transcription automatique du chapitre 7 et la partition de musique dans un format numérique unique bien connu du système. Bien que la couche de transcription automatique ait détecté plusieurs notes potentielles, à cette étape de traitement, le module ne sait toujours pas à quelles notes de la partition correspondent celles qui sont correctement détectées. Ainsi, c'est le rôle de la couche d'alignement de faire ce *mapping* comme démontré dans l'exemple de la figure 8.1 où quatre notes potentielles correctement détectées sont alignées à quatre notes de la partition. Ce *mapping* que l'on nomme *Alignement* est l'unique sortie de cette couche.

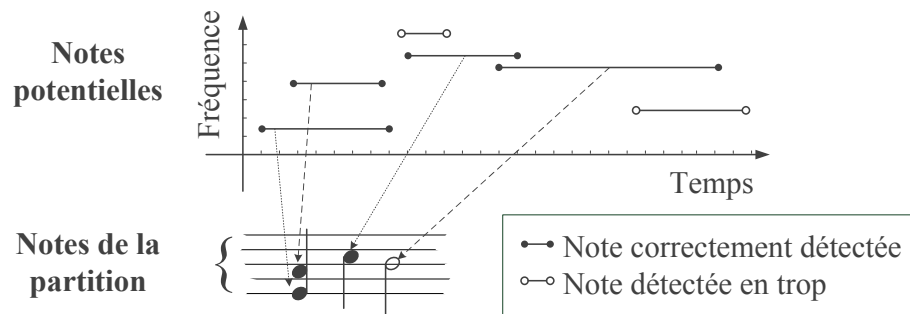


Figure 8.1 Exemple d'alignement

Ce *mapping* est utile pour répondre à des questions importantes telles que :

1. Après x secondes, quelle mesure et quelle(s) note(s) jouait le musicien ?
2. Quel extrait de la partition de musique jouait le musicien ?
3. Quel était le tempo en battement par minutes à la mesure x ?

D'ailleurs, lorsque l'alignement est effectué en continu, il peut être utilisé pour produire un accompagnement automatique, et ce, à partir d'une application qui joue aux bons moments les notes de la partition d'accompagnement pendant que le musicien joue, de son côté, les notes de sa partition. Il est à noter que la couche d'alignement discutée dans ce chapitre est implémentée de façon à permettre le fonctionnement en continu, et, par conséquent, peut

être utilisée pour réaliser ce type d'application. D'ailleurs, cette implémentation a aussi pour objectif de pouvoir éventuellement répondre à la première portion de la question de recherche initialement présentée à l'introduction.

8.1 Aperçu du système d'alignement proposé

Tel qu'illustré à figure 8.2, la couche d'alignement est réalisée par deux blocs de traitement, c'est-à-dire le bloc d'alignement brut suivi du bloc d'alignement fin. L'alignement brut permet de faire rapidement un premier *mapping* en recherchant à travers toute la partition de musique les extraits pouvant avoir été joués par le musicien. Afin de réduire le temps de traitement, l'alignement brut utilise essentiellement en entrée les notes les plus probables des notes potentielles détectées par la couche de transcription automatique du chapitre 7. L'alignement fin est ensuite utilisé pour effectuer un *mapping* plus précis en puisant dans toute la liste de notes potentielles. Contrairement à l'alignement brut, l'alignement fin est beaucoup plus précis et, par conséquent, davantage demandant en terme de quantité de traitement. Aussi, afin de simplifier sa tâche, l'alignement fin limite sa recherche à l'intérieure de bornes temporelles définies par l'alignement brut. Il est à noter que le bloc d'alignement brut génère en sortie plusieurs alignements potentiels. Dans les faits, un seul de ces alignements potentiels devrait correspondre à ce qui a réellement été joué par le musicien. C'est aussi le rôle de l'alignement fin de sélectionner parmi tous ces candidats l'alignement le plus probable. Les deux prochaines sections présentent en détail le fonctionnement de chacun de ces blocs d'alignements.

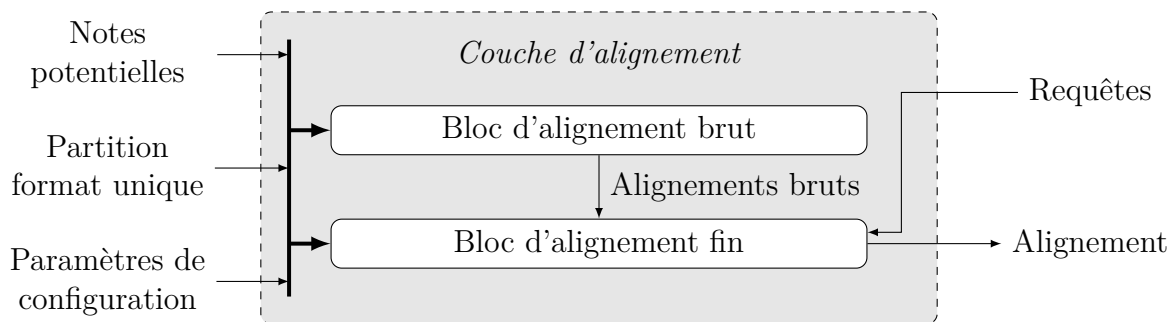


Figure 8.2 Schéma bloc de la couche d'alignement

8.2 Alignement brut

Contrairement aux algorithmes d'alignement habituels [2, 8, 11, 12, 28, 39, 43, 54, 63, 64, 65, 69, 70, 72, 73, 76], l'alignement brut donne la liberté au musicien de commencer à jouer là où il le désire dans la partition de musique, et ce, sans avoir à préalablement informer le système de ses intentions. Le musicien peut aussi sauter d'un endroit à l'autre dans la partition de musique. Bien entendu, pour obtenir de bons résultats, le musicien doit jouer

des segments suffisamment longs pour permettre à l’alignement brut de bien identifier ces derniers.

L’alignement brut s’inspire de la technique d’alignement *Dynamic Time Warping* (DTW) et plus particulièrement de l’algorithme d’alignement monophonique *Extensive Dynamic Time Warping* (EDTW) [22] développé lors de la maîtrise de l’auteur de cette thèse qui est pour sa part un algorithme dérivé du DTW. De ce fait, afin de faciliter la compréhension du fonctionnement de l’algorithme d’alignement brut proposé, il est jugé pertinent d’avoir préalablement parcouru les sections 3.3 et 3.4 du chapitre 3 résumant le fonctionnement des algorithmes DTW et EDTW ainsi que les divers éléments utilisés par l’algorithme EDTW tels que la matrice de similarité, les contraintes de déplacement et les chemins survivants. La sous-section suivante présente les différents éléments de l’algorithme d’alignement brut proposé ainsi que les principales différences par rapport à l’algorithme original d’alignement EDTW.

8.2.1 Éléments de l’algorithme

L’algorithme proposé est en réalité une importante évolution de l’algorithme EDTW. Bien qu’en surface, l’algorithme d’alignement proposé puisse ressembler à l’algorithme EDTW, au niveau de la mécanique interne, il existe d’importantes différences. Ceci est dû au fait que contrairement à l’algorithme EDTW, l’algorithme proposé a été développé pour fonctionner avec des partitions de musique contenant des notes jouées en polyphonie, c’est-à-dire des notes dont une portion de leur durée doit être jouée en même temps qu’une ou plusieurs autres notes. À titre d’exemple, la figure 8.3 présente un extrait d’une partition où les trois notes dans la zone ombragée doivent être jouées en polyphonie¹.

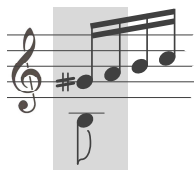


Figure 8.3 Exemple des notes jouées en polyphonie

1. Effectivement, les deux premières notes du haut (Sol# et La), devant être jouées l’une après l’autre dans le temps, possèdent individuellement une durée deux fois plus courte que la première note du bas (Si) et, par conséquent, la note du bas démarrant au même instant que la première note du haut doit être maintenu pendant toute la durée de l’exécution des deux premières notes du haut. Finalement, comme la première note du bas est terminée, les deux dernières notes doivent être jouées seules l’une à la suite de l’autre, c’est-à-dire de façon monophonique.

Axes de la matrice de similarité représentant la partition de musique

La figure 8.4 présente un exemple fictif d'extrait de matrice de similarité utilisée par l'algorithme modèle, c'est-à-dire l'algorithme EDTW. Pour cet algorithme, les éléments de chacun des deux axes de la matrice de similarité sont des notes de musique agencées selon leur ordre d'apparition dans le temps. Plus précisément, les éléments de ces deux axes correspondent aux notes de la partition et aux notes détectées pour lesquelles sont rattachés une durée, un moment de début et un moment de fin. Chacune

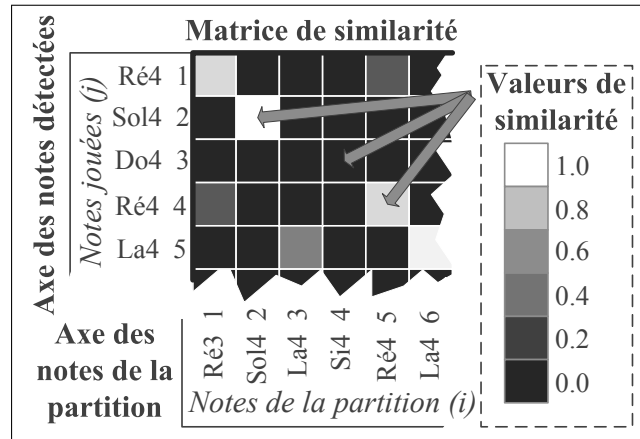


Figure 8.4 Exemple fictif de matrice de similarité utilisée par l'algorithme EDTW

des valeurs de similarité correspond à une valeur proportionnelle à la ressemblance entre la note jouée j et la note de la partition i , et ce, au niveau de leur identité, durée et moment de début. L'agencement selon l'ordre d'apparition dans le temps permet d'établir les contraintes de déplacement de la figure 8.5, et ce, avec l'hypothèse simplificatrice que le musicien ne jouera jamais la partition de musique à reculons.

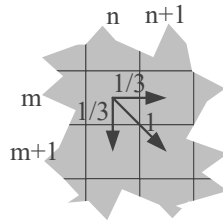


Figure 8.5 Contraintes de déplacement de l'algorithme EDTW

Au niveau polyphonique, comme plusieurs notes peuvent démarrer au même instant, la convention suivante a été utilisée : regrouper toutes les notes commençant au même instant dans un même élément. Ainsi, les éléments de l'axe de la partition pour les notes de la figure 8.3 seraient agencés selon le tableau 8.1.

Tableau 8.1 Axe d'index (i) de la partition correspondant à l'ensemble des notes du segment présenté à la figure 8.3

Index (i)	1	2	3	4
Ensemble de notes	Si3 Sol#4	La4	Si4	Do5

Au niveau de l'axe vertical représentant les notes détectées, le même principe est appliqué. C'est-à-dire qu'un élément peut représenter un ensemble de notes démarrant au même instant ou une note démarrée seule. Malencontreusement, comme le musicien n'est pas parfait, certaines notes devant commencer au même instant peuvent être décalées des autres de quelques millisecondes. Afin d'illustrer ceci, considérons l'exemple fictif de la figure 8.6 représentant sept notes potentielles détectées par la couche de transcription automatique. En admettant que les notes (A) et (B) soient séparées entre elles par un délai de seulement quelques dizaines de millisecondes, comment pouvons-nous affirmer avec certitude que l'intention du musicien était bel et bien de jouer ces notes l'une à la suite de l'autre ? Peut-être que, selon la partition de musique, il s'agissait plutôt de deux notes devant être jouées en simultanée. Comme il est impossible d'en être certain, l'axe vertical de la matrice de similarité doit permettre ces deux cas.

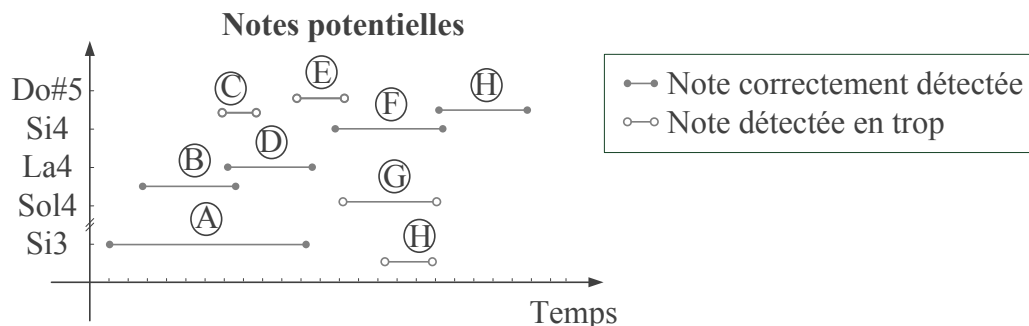


Figure 8.6 Exemple de notes potentielles utilisées pour créer l'axe d'index de note à départ commun au niveau des notes détectées

La première étape de construction de l'axe consiste à repérer toutes les notes potentielles respectant les conditions suivantes :

1. Intensité suffisamment élevée ($loud(p, X) > 11.2$ selon l'équation 7.2) ;
2. Niveau de confiance suffisamment élevée ($c_a > 0.65$ selon l'équation 7.4) ;
3. Niveau de confiance de l'octave suffisamment élevée ($c_n > 0.4$ selon l'équation 7.11) ;
4. Identité (ex. Si3) existante dans la partition de musique.

Pour toutes fins pratiques, ces notes sont nommées « notes déclencheurs » puisqu'elles ont pour fonction de déterminer des positions dans le temps où il s'avère très probable d'y trouver un événement sonore réel, soit le début de plusieurs notes jouées en polyphonie ou le début d'une note jouée en monophonie. Afin d'illustrer le fonctionnement de la construction de l'axe, les notes déclencheurs de la figure 8.6 sont identifiées par une ligne plus épaisse à la figure 8.7.

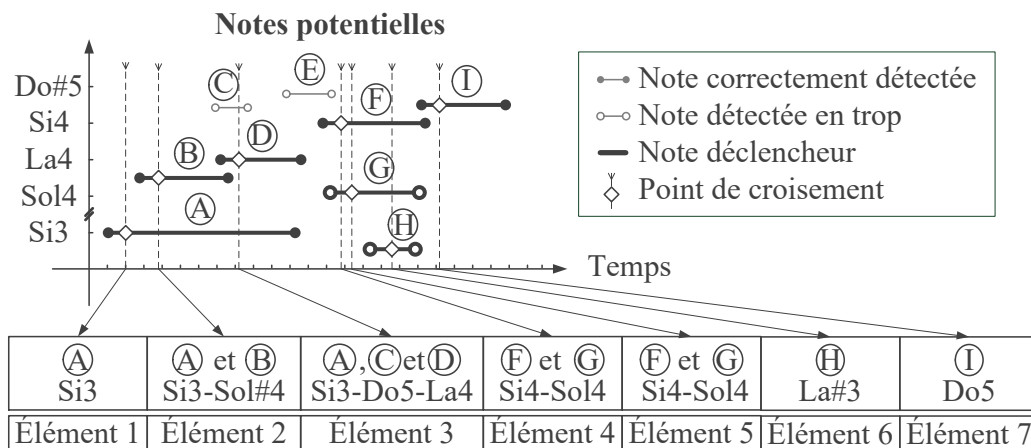


Figure 8.7 Exemple d'axe d'index (j) d'un ensemble de notes à départ commun au niveau des notes détectées

La deuxième étape consiste à positionner des points nommés « points de croisement » à droite de chacune des notes déclencheurs tels qu'illustrés à la figure 8.7. Afin d'obtenir de bons résultats, ces derniers doivent être positionnés à environ 50 millisecondes à droite de la note déclencheur en question. Il est à noter que cette valeur correspond à la distance entre deux notes consécutives jouées à très haute vitesse, c'est-à-dire 20 notes par secondes.

La troisième étape consiste à regrouper ensemble dans un élément de l'axe toutes les notes potentielles chevauchant le même point de croisement tel qu'illustré à la figure 8.7. Pour terminer, la dernière étape consiste à retirer tous les doublons consécutifs, c'est-à-dire les éléments identiques à un autre élément adjacent. Par exemple, pour le cas de la figure 8.7, le cinquième élément serait retiré comme il est identique au quatrième élément étant composé des notes Si4 et Sol4.

Toutes ces étapes permettent de créer un axe contenant plusieurs possibilités d'agencement de notes. Par exemple, comme il n'est pas possible d'affirmer avec certitude si les notes Ⓐ et Ⓑ sont des notes devant démarrer au même instant ou des notes devant démarrer une à la suite de l'autre, les deux cas font partie de l'axe, c'est-à-dire l'élément 2 pour le premier cas et l'élément 1 et 2 pour le deuxième cas. Bien que, pour le deuxième cas, l'élément 2 contienne aussi la note Ⓐ, il sera démontré à la sous-section suivante que certaines des notes contenues dans un élément peuvent être ignorées lors du calcul des valeurs de similarité.

Il est à noter que bien que la note Ⓐ semble avoir démarré trop tôt pour être regroupée avec la note Ⓓ, l'élément 3 a tout de même été créé. En fait, cet élément est utile pour compenser une imperfection de la couche de transcription automatique du chapitre 7.

En effet, il arrive assez fréquemment que cette dernière couche détecte une note trop tôt lorsque deux notes de même tonie sont jouées consécutivement avec une transition douce. Ainsi, il pourrait arriver que la note \textcircled{A} soit en réalité composée de deux notes consécutives de même tonie et que la note, inadéquatement jumelée à droite de \textcircled{A} , débutât approximativement au même instant que la note \textcircled{D} .

Le fait de créer les éléments de l'axe qu'avec des notes très probables, c'est-à-dire qu'avec les notes déclencheurs, permet de réduire de façon considérable le nombre d'éléments dans l'axe comparativement au cas où chacune des notes créerait un élément. Finalement, on pourrait se demander pourquoi les points de croisement n'ont pas été directement positionnés sur les notes déclencheurs. En fait, ce léger décalage à droite permet d'éviter de regrouper ensemble des notes qui, de façon évidente, ne devaient pas être démarrées au même instant. C'est-à-dire des notes où le chevauchement est minime, tel que c'est le cas pour les notes \textcircled{B} et \textcircled{D} .

Matrice et ses valeurs de similarité

Il est à noter que pour l'algorithme d'alignement brut, le calcul des valeurs de similarité a été simplifié par rapport à l'algorithme EDTW. En fait, la complexité de calcul de l'algorithme EDTW a simplement été déplacée vers une autre étape de traitement, c'est-à-dire la construction des chemins survivants discutée à la sous-section 8.2.2.

Pour l'alignement brut, la valeur de similarité d'un élément de coordonnées (i,j) correspond simplement au rapport suivant : le nombre de notes dans l'élément (j) qui sont de même identité que les notes dans l'élément (i) sur le nombre de notes dans l'élément (i) . Pour illustrer ceci à l'aide d'un exemple, la matrice de similarité de la figure 8.8 est présentée pour le cas où les deux axes de cette matrice correspondraient aux éléments du tableau 8.1 et de la figure 8.7. Par exemple, l'élément de coordonnées $(1,1)$ donne une valeur de similarité de 0.5 (rapport $1/2$) puisqu'il y a, dans l'élément $j=1$, qu'une seule note de même identité que les deux notes de l'élément $i=1$, c'est-à-dire la note Si3.

Chemins survivants et contraintes de déplacement

Un chemin survivant est en quelque sorte un segment potentiel d'alignement qui est composé de plusieurs valeurs de similarité non nulles connectées entre elles selon des contraintes bien précises telles que les contraintes de déplacement de la figure 8.5. La figure 3.11(a) présente un exemple concret de chemins survivants obtenus à l'aide de l'algorithme EDTW, et ce, à partir de la matrice de similarité de la figure 3.10.

Imaginons que l'algorithme EDTW soit appliqué sur la matrice de similarité de la figure 8.8 et que l'algorithme ait retenu les deux chemins survivants de la figure 8.9, et ce, en utilisant les contraintes de déplacement de la figure 8.5. Pour ce cas, l'algorithme EDTW interpré-

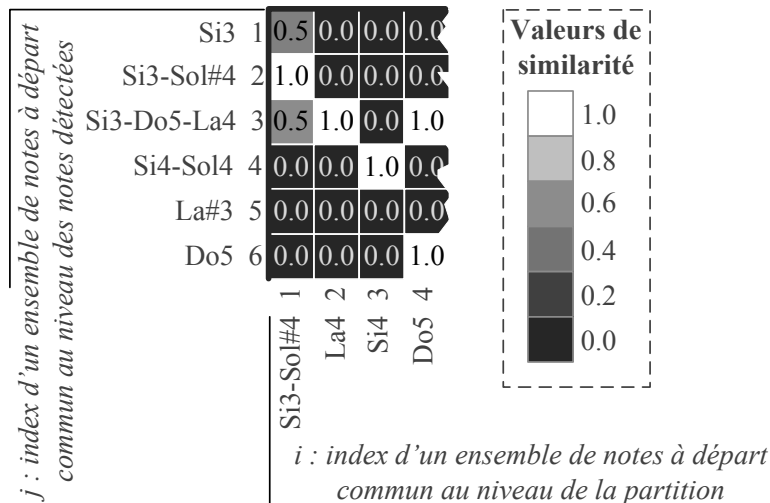


Figure 8.8 Exemple de matrice de similarité utilisé par l'alignement brut

terait le premier chemin survivant comme suit : la deuxième note détectée, c'est-à-dire l'élément 2 de l'axe vertical, correspond à la première note de la partition, la troisième note jouée correspond à la première et deuxième note de la partition, la quatrième note jouée correspond à la troisième note de la partition.

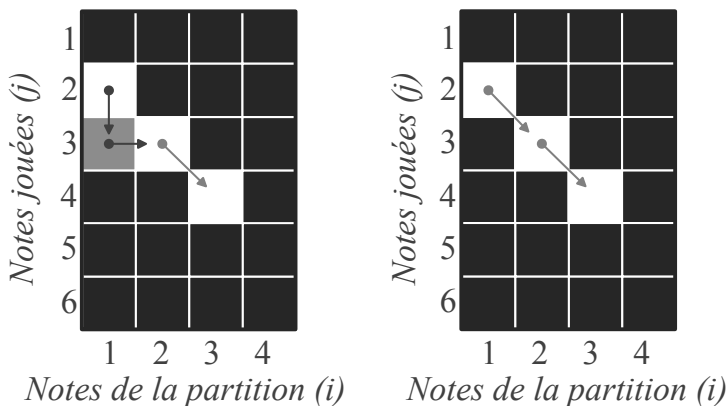


Figure 8.9 Exemple de chemins survivants pour l'algorithme EDTW

Au niveau polyphonique, ces affirmations sont complètement incohérentes puisque certains éléments des axes du tableau 8.1 et de la figure 8.7 sont composés de plusieurs notes et puisque les contraintes de déplacements de la figure 8.5 ont été conçues sous l'hypothèse que toutes les notes ont été détectées correctement, c'est-à-dire qu'il ne manque aucune note et qu'il n'y a pas de notes en extra. En polyphonie, avec les algorithmes de traitement du signal actuels, il n'est pas possible d'arriver à une telle perfection. C'est d'ailleurs pour cette raison que les notes détectées par la couche de transcription automatique du

chapitre 7 sont des notes potentielles et, par conséquent, des notes dont l'existence réelle est incertaine.

De ce fait, les contraintes de déplacement de l'algorithme EDTW ont été remplacées par les contraintes de déplacement de la figure 8.10. Il est intéressant de remarquer que les nouvelles contraintes de déplacement acceptent maintenant les sauts de plusieurs éléments. Les sauts verticaux supérieurs à un permettent de tenir compte du fait que certaines notes potentielles peuvent être des notes détectées en extra, c'est-à-dire des notes que le musicien n'a pas jouées, mais que la couche de transcription a tout de même détectées par erreur. Quant aux sauts horizontaux supérieurs à un, ils permettent de tenir compte du fait que parfois certaines notes jouées par le musicien peuvent ne pas avoir été détectées par la couche de transcription automatique. Contrairement aux sauts à la verticale, la distance maximale du saut acceptée à l'horizontale est de deux, c'est-à-dire de n à $n+2$. Ceci s'explique par le fait que la couche de transcription automatique a été conçue pour détecter presque toutes les notes jouées, et ce, en acceptant un taux non négligeable de notes détectées en extra. En effet, la probabilité qu'une note ne soit pas détectée est beaucoup plus faible que celle qu'une note soit détectée en trop, c'est-à-dire environ une non-détection pour vingt détections en extras. Des essais réalisés avec plusieurs pièces de violon ont permis de démontrer qu'un saut horizontal maximum de deux ainsi qu'un saut vertical maximum de dix permettaient d'obtenir de bons résultats d'alignement.

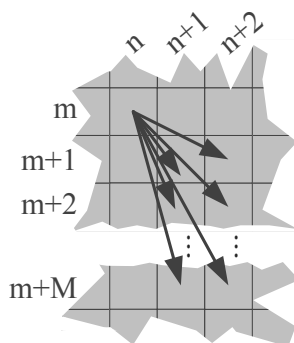


Figure 8.10 Contraintes de déplacement de l'alignement brut

Il est aussi intéressant de remarquer que les contraintes de déplacements purement vertical et horizontal de l'algorithme EDTW n'ont pas été conservées, soit lorsque l'objectif du déplacement vertical était de détecter les notes de la partition répétées par erreur par le musicien et l'objectif du déplacement horizontal était de détecter les notes omises par le musicien. Ce choix s'explique par le fait que l'objectif de l'alignement brut n'est plus de trouver les erreurs de performance du musicien, mais plutôt de faire un *mapping* approxi-

matif de ce qui a été joué afin de réduire le temps de traitement nécessaire pour réaliser l'étape ultérieure, c'est-à-dire l'alignement fin.

Contrairement aux contraintes de déplacement de l'alignement brut de la figure 8.10, chacun des déplacements de l'algorithme EDTW de la figure 8.5 est accompagné d'un coefficient de pondération, c'est-à-dire une valeur de $1/3$ pour les déplacements vertical et horizontal, et une valeur de 1 pour le déplacement oblique. Ces coefficients permettaient de pondérer, selon le déplacement, chacune des valeurs de similarité contenue dans le chemin survivant. Plus précisément, ces coefficients permettaient de défavoriser les déplacements purement vertical et horizontal, comme ces derniers sont jugés occasionnels par le fait qu'ils reflètent des erreurs de performance commises par le musicien. La section 3.4 du chapitre 3 explique en plus de détails l'utilité de ce coefficient.

Pour l'algorithme d'alignement brut, le coefficient de pondération a été retiré puisque tous les déplacements sont jugés probables et, donc, aucun d'eux ne devrait être pénalisé simplement par un jugement sur la direction. En fait, pour l'alignement brut, la pondération d'une direction par rapport à une autre est calculée selon des critères basés sur des attributs musicaux. Afin d'aider à la compréhension, imaginons les deux chemins survivants fictifs de la figure 8.11 respectant les contraintes de déplacement de la figure 8.10.

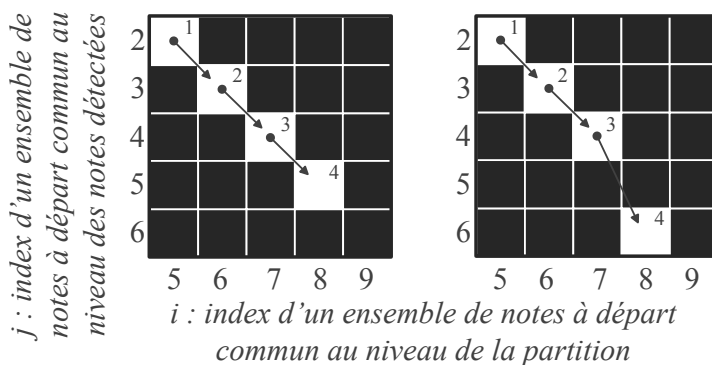


Figure 8.11 Exemple fictif de chemins survivants

D'abord, à partir des données des tableaux 8.2 et 8.3 correspondants aux débuts des notes associées aux quatre éléments des deux chemins survivants de la figure 8.11, il est possible de calculer le tempo (rythme musical) impliqué par les déplacements à travers la matrice. Par exemple, pour les trois premiers éléments, c'est-à-dire les éléments communs aux deux chemins survivants, le tempo r en secondes par mesures (SPM) pourrait être évalué à l'aide de l'équation 8.1.

Tableau 8.2 Début des notes pour chacun des éléments contenus dans le premier chemin survivant de la figure 8.11

		1	2	3	4
Axe des notes détectées	j	2	3	4	5
	Début moyen (secondes)	$x_1 = 3.7$	$x_2 = 4.6$	$x_3 = 5.1$	$x_4 = 5.3$
Axe des notes de la partition	i	5	6	7	8
	Début (mesures)	$y_1 = 2.0$	$y_2 = 2.5$	$y_3 = 2.75$	$y_4 = 3.0$

Tableau 8.3 Début des notes pour chacun des éléments contenus dans le deuxième chemin survivant de la figure 8.11

		1	2	3	4
Axe des notes détectées	j	2	3	4	6
	Début moyen (secondes)	$x_1 = 3.7$	$x_2 = 4.6$	$x_3 = 5.1$	$x_4 = 5.6$
Axe des notes de la partition	i	5	6	7	8
	Début (mesures)	$y_1 = 2.0$	$y_2 = 2.5$	$y_3 = 2.75$	$y_4 = 3.0$

$$r = \text{mean} \left(\frac{x_2 - x_1}{y_2 - y_1}, \frac{x_3 - x_2}{y_3 - y_2} \right) \quad (8.1)$$

$$r = (1.8 + 2.0) / 2 = 1.9 \text{ (secondes par mesures)}$$

À partir de cette information, il est possible d'évaluer le moment de début en secondes attendu pour le quatrième élément. En effet, le tableau 8.2 permet d'affirmer que le groupement de notes à la position $i = 8$ commence 0.25 mesure à la suite de l'élément à la position $i = 7$, par conséquent, le moment de début attendu pour le prochain groupement de notes est donc de 5.575 secondes, soit $1.9(SPM) \times 0.25(\text{mesures}) + 5.1(s)$.

Comme la valeur $x_4 = 5.6$ du tableau 8.3 s'avère plus rapprochée de la valeur 5.575, et ce, par rapport à la valeur $x_4 = 5.3$ du tableau 8.2, il est possible d'affirmer qu'à partir de cette information, le quatrième élément du deuxième chemin survivant semble beaucoup plus probable que le quatrième élément du premier chemin survivant, et ce, malgré la présence d'un saut supérieur à 1. Ainsi, comme mentionné précédemment, plutôt que de pénaliser un type de déplacement par le biais d'un coefficient de pondération, l'algorithme

d'alignement brut utilise plutôt des calculs basés sur des critères musicaux pour évaluer un déplacement par rapport à un autre. Plus précisément, chacun de ces critères permet de vérifier la cohérence d'un chemin survivant relativement au respect de certaines des règles de la théorie de la musique telles que le respect du rythme, de la mélodie, de l'articulation² et du tempo.

Voici la liste des critères musicaux utilisés pour évaluer les déplacements :

1. Respect du rythme :

Proximité de la position de début attendue (d_a) par rapport à celle obtenue (d_o) pour l'élément ajouté (l);

2. Respect de l'articulation :

Proximité des durées attendues par rapport à celles obtenues pour les notes contenues dans l'élément ajouté;

3. Constance du tempo :

Consistance du tempo impliqué par l'élément ajouté par rapport à celui des éléments précédents;

4. Respect du tempo :

Proximité du tempo prescrit dans la partition par rapport à celui obtenu pour l'élément ajouté;

5. Respect de la mélodie :

Intensité des notes potentielles contenues dans le chemin survivant par rapport à celles qui ont été sautées.

Tout comme pour les tonies potentielles trouvées par la couche de traitement du signal du chapitre 6 et les notes potentielles détectées par la couche de transcription automatique du chapitre 7, chacun de ces critères musicaux possède un niveau de confiance ayant une valeur dans la plage $[0,1]$. Par exemple, pour le premier critère musical de la liste, le niveau de confiance $c_1(k, l)$ associé à l'élément l d'un chemin survivant k est calculé selon l'équation 8.2. Il est à noter que l'annexe C présente tous les calculs permettant d'obtenir les cinq niveaux de confiance (c_1, c_2, \dots, c_5) associés à tous les critères musicaux listés ci-dessus.

$$c_1(k, l) = \begin{cases} 1.0, & l \leq 2 \\ \max(0.0, 1.0 - |d_a(k, l) - d_o(k, l)|, 0) \end{cases} \quad (8.2)$$

2. En musique, l'articulation permet de préciser la durée des mini-silences entre les notes et la façon d'attaquer ces dernières.

Les chemins survivants d'indice k possèdent eux aussi un niveau de confiance ($c_s(k)$) compris dans la plage $[0,1]$. Ce niveau de confiance est obtenu à l'aide de l'équation 8.3. Il est utilisé lors de l'étape de sélection des meilleurs chemins survivants, et ce, afin de produire l'alignement de sortie de l'algorithme.

$$c_s(k) = \frac{1}{5L} \sum_{l=1}^L \sum_{i=1}^5 c_i(k, l) \quad (8.3)$$

Il est à noter que pour l'algorithme EDTW, le calcul des valeurs de similarité était basé sur certains de ces critères musicaux. En fait, l'algorithme EDTW approximait un tempo pour une valeur de similarité donnée et effectuait ensuite le calcul de cette valeur de similarité en fonction de divers critères musicaux. Le fait d'effectuer maintenant ce type de calculs au niveau du chemin survivant permet d'éviter l'étape d'approximation du tempo fait par l'algorithme EDTW et, ainsi, permet d'augmenter la performance de l'algorithme d'alignement brut par rapport à l'algorithme EDTW.

Alignements bruts

L'algorithme d'alignement brut permet de générer en sortie plusieurs alignements potentiels nommés alignements bruts. Chacun de ces alignements est constitué de plusieurs chemins survivants fusionnés ensemble et, tout comme pour le chemin survivant, il peut être représenté graphiquement à l'intérieur de la matrice de similarité. Le choix des chemins survivants constituant l'alignement est effectué selon des contraintes bien précises discutées à la prochaine sous-section. Parmi tous les alignements bruts trouvés par l'algorithme, si tout s'est bien passé, l'un d'entre eux devrait représenter ce qui a été réellement joué par le musicien. C'est d'ailleurs le rôle du bloc d'alignement fin de la figure 8.2 et discuté à la prochaine section d'identifier cet alignement parmi tous les autres alignements.

Fenêtre adaptative de calcul (FAC)

Une partition de musique de plusieurs pages peut contenir plusieurs milliers de notes et, par conséquent, peut produire une matrice de similarité constituée de plusieurs milliers de colonnes. À priori, comme l'algorithme d'alignement brut ne sait pas quels segments de la partition de musique seront joués par le musicien, l'algorithme doit calculer tous les éléments de la matrice de similarité et, ensuite, rechercher à travers cette énorme matrice tous les chemins survivants possibles, et ce, afin de déceler les meilleurs candidats pour produire les alignements de sorties potentiels. Évidemment, l'accomplissement de tous ces calculs peut prendre un temps considérable qui peut rendre le traitement en temps réel impossible, c'est-à-dire le rafraîchissement rapide et continu des alignements bruts potentiels, et ce, tout le long du jeu du musicien.

Bien entendu, au bout d'un certain temps, le segment joué par le musicien sera probablement identifié par l'algorithme d'alignement. Ainsi, une fois identifié, l'algorithme d'alignement brut pourrait se permettre d'omettre le calcul de certaines colonnes de la matrice de similarité éloignées de l'endroit joué par le musicien, et ce, pour permettre le fonctionnement en continu nécessaire à certaines applications telles que l'accompagnement automatique. Pour ce faire, un nouvel élément a été ajouté à l'algorithme, c'est-à-dire la fenêtre adaptative de calcul (FAC). À titre d'exemple, la figure 8.12 présente un extrait d'alignement à l'intérieur d'une matrice de similarité dont certaines valeurs de similarité n'ont pas été calculées puisqu'elles étaient à l'extérieur de la FAC.

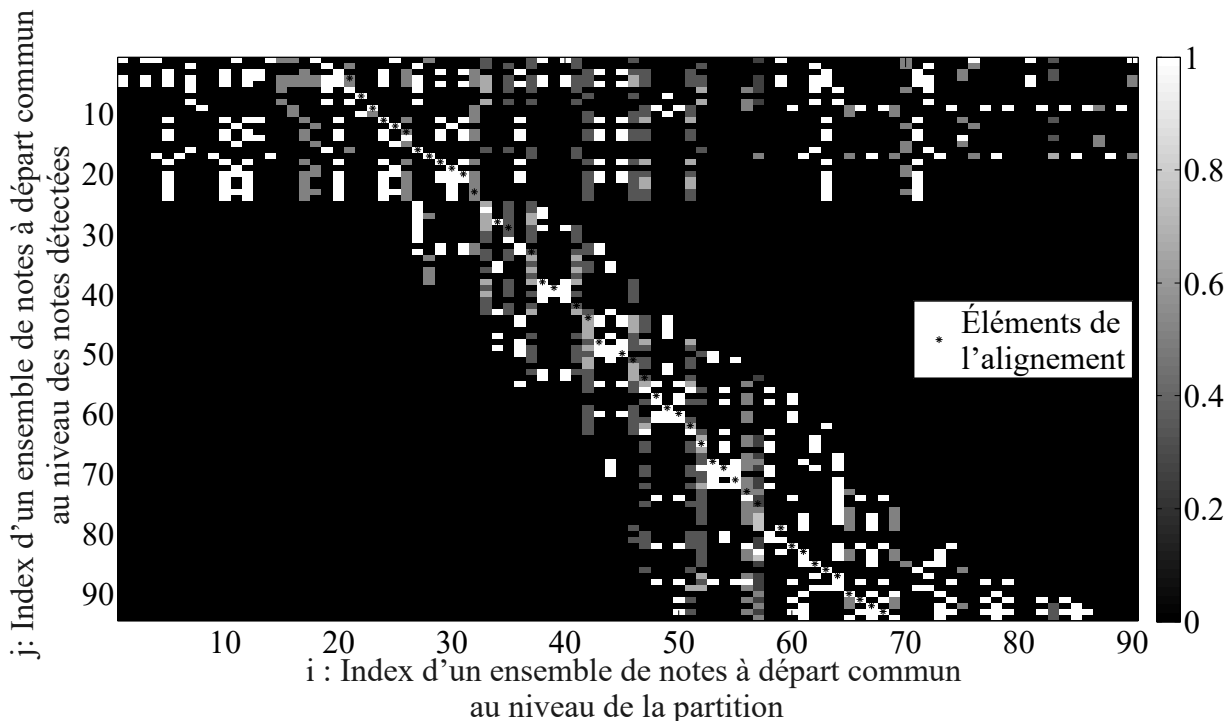


Figure 8.12 Exemple d'alignement obtenu à l'aide de la fenêtre adaptative de calcul

Ainsi, pour une ligne donnée de la matrice de similarité, la fenêtre adaptative de calcul a pour objectif de réduire le nombre de colonnes à calculer. Elle est composée d'une borne inférieure et supérieure qui s'ajuste de façon adaptative, et ce, en fonction de la ligne de la matrice et de l'alignement obtenu à la sortie du bloc d'alignement fin. Plus précisément, toutes les valeurs de similarité dont l'index de l'élément de l'axe de la partition est en dessous de la borne inférieure et au-dessus de la borne supérieure ne sont pas calculées. Il est à noter que la FAC peut être désactivée pour les cas où la dimension de la partition est suffisamment petite pour permettre le fonctionnement en temps réel de type meilleur effort, et ce, en prenant en compte l'ensemble de la pièce musicale.

Le caractère adaptatif de la fenêtre est dû au fait que le nombre d'éléments calculés diminue progressivement lorsque l'algorithme d'alignement estime avoir trouvé le segment joué par le musicien et, pour le cas contraire, le nombre d'éléments calculés augmente progressivement tant que l'algorithme n'estime pas avoir retrouvé le segment joué par le musicien. Ainsi, en évitant de faire des calculs inutiles, la fenêtre adaptative de calcul permet de libérer des ressources CPU pour ainsi assurer un fonctionnement en temps réel de type meilleur effort.

8.2.2 Étapes de fonctionnement

Suite à la présentation des divers éléments de l'algorithme d'alignement brut, il est maintenant possible de présenter les diverses étapes de traitement telles qu'illustrées à la figure 8.13. Il est à noter que ces étapes permettent d'obtenir plusieurs alignements bruts potentiels en continu. Pour ce faire, les étapes 1 à 6 sont continuellement répétées, et ce, tant et aussi longtemps que l'algorithme est actif.

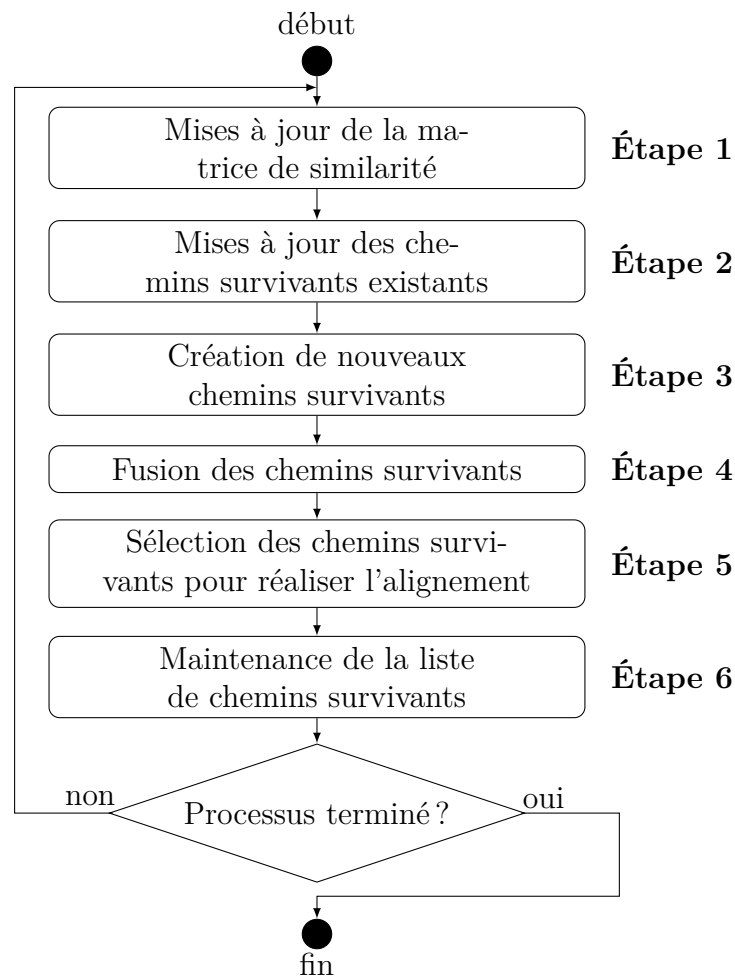


Figure 8.13 Différentes étapes de l'alignement brut

Étape 1 : Mise à jour de la matrice de similarité

Tout au long du jeu du musicien, la couche de transcription automatique ajoute des notes potentielles à la liste. La première étape de l'algorithme d'alignement brut consiste à mettre à jour l'axe vertical des notes détectées, et ce, en utilisant les notes potentielles nouvellement ajoutées à la liste. Une fois la mise à jour de l'axe terminée, la matrice de similarité est à son tour mise à jour. Il est à noter que ces mises à jour sont effectuées selon les techniques discutées à la sous-section 8.2.1.

Sans ajout supplémentaire, le nombre de lignes de la matrice augmenterait continuellement. Cependant, pour éviter de remplir inutilement la mémoire physique, seuls les éléments de l'axe vertical récents sont conservés tels qu'illustrés à la figure 8.14. D'ailleurs, pour le système conçu et pour la figure 8.14, seuls les éléments datant de moins de 15 secondes sont conservés.

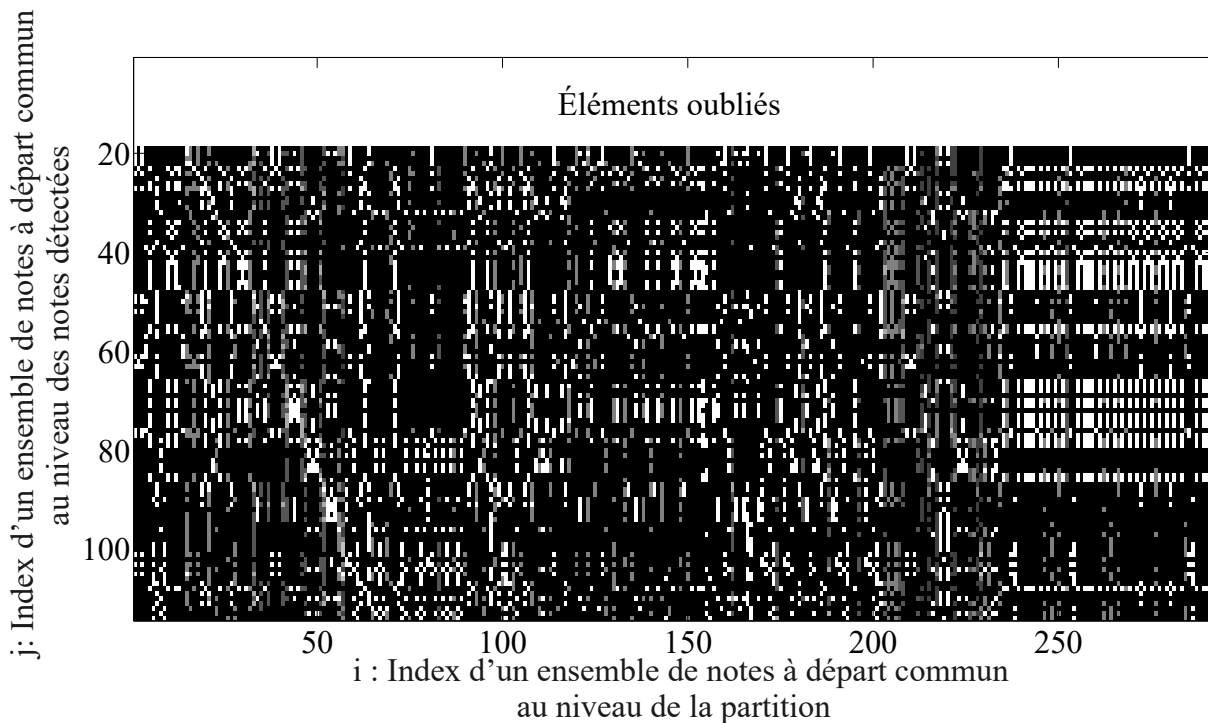


Figure 8.14 Exemple de matrice de similarité avec des éléments oubliés

Comme discuté à la sous-section 8.2.1, lorsque la fenêtre adaptative de calcul (FAC) est utilisée, seuls les éléments de la matrice à l'intérieure de cette fenêtre sont calculés tels qu'illustrés à la figure 8.12. Bien que la largeur de base de cette fenêtre ne soit ajustée qu'une fois l'alignement de sortie trouvé par le bloc d'alignement fin, le centre de la fenêtre est, quant à lui, ajusté à chacune des lignes de la matrice de similarité mise à jour lors de cette première étape. Cet ajustement permet de tenir compte du fait que l'alignement se

déplace en pente descendante tel que vue pour l'exemple de la figure 8.12. En effet, comme le musicien ne joue logiquement pas la partition de musique à reculons, l'alignement doit nécessairement prendre la forme d'une pente descendante.

L'alignement de sortie du bloc d'alignement fin est utilisé pour initialiser la position de la fenêtre adaptative de calcul (FAC). Lors de cette initialisation, le centre de la FAC est positionné sur le dernier élément de l'alignement tel qu'illustré à la figure 8.15. Par contre, lors de la toute première exécution de l'étape 1, la FAC est initialisée pour couvrir toutes les colonnes de l'axe de la partition ou, si spécifiée par l'utilisateur, initialisée pour couvrir les mesures de la partition ou le musicien commencera normalement à jouer.

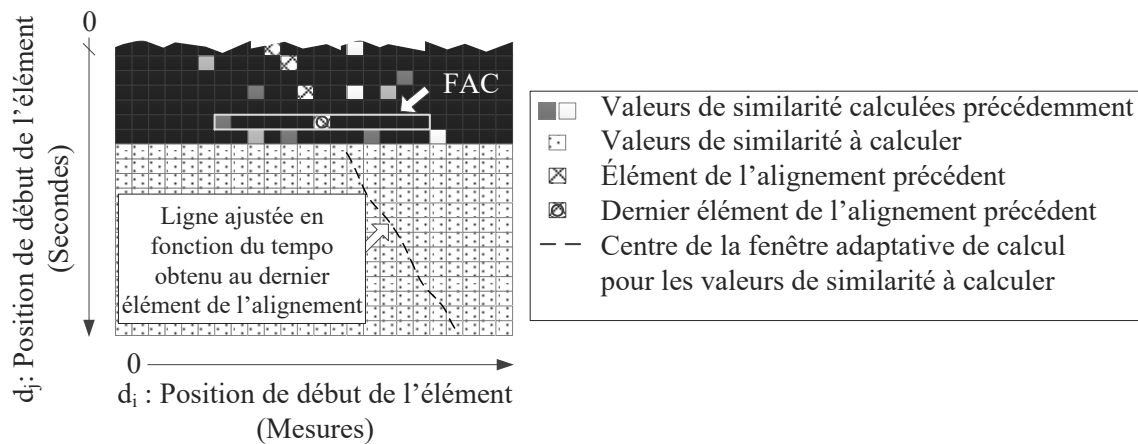


Figure 8.15 Exemple de gabarit pour positionner le centre de la fenêtre adaptative de calcul

À l'intérieur de la matrice de similarité fictive de la figure 8.15, la position du centre de la FAC pour les lignes en dessous du dernier élément de l'alignement est représentée par la ligne en pointillé. De plus, les index i et j habituels de l'axe ont été remplacés par la position de début de ces éléments, c'est-à-dire d_i en mesures pour l'axe de la partition et d_j en secondes pour l'axe des notes détectées.

Afin de positionner le centre de la FAC, et ce, pour chacune des lignes de la matrice à calculer, le tempo $\tau[k]$ obtenu en mesures par seconde au dernier élément ($k = K$) de l'alignement de sortie est d'abord calculé à l'aide de l'équation 8.4 où $\hat{d}_j[k]$ et $\hat{d}_i[k]$ correspondent à la position de début en secondes de l'élément j de l'axe vertical et à la position de début en mesure de l'élément i de l'axe horizontal respectivement, et ce, pour le k -ième élément de l'alignement.

$$\tau[k] = \frac{1}{N+1} \sum_{n=0}^N \left(\frac{\hat{d}_i[k-n] - \hat{d}_i[k-n-1]}{\hat{d}_j[k-n] - \hat{d}_j[k-n-1]} \right) \quad (8.4)$$

Finalemment, le numéro de mesure α_j correspondant au centre de la FAC pour une ligne donnée j de la matrice est obtenu à l'aide de l'équation 8.5 où d_j correspond à la position de début en secondes de la ligne j dont il est question. En quelques mots, ce centre correspond à la prédiction de l'élément i de l'axe de la partition pour une ligne donnée j qui ferait partie de l'alignement subséquent, et ce, sous l'hypothèse que le musicien jouerait le même segment de partition de musique que celui joué lors du dernier alignement, et ce, en conservant un tempo constant.

$$\alpha_j = \hat{d}_i[K] + (d_j - \hat{d}_j[K]) \tau(K) \quad (8.5)$$

Construction et mémorisation d'un chemin survivant

Les étapes 2 et 3 de la figure 8.13 partagent toutes les deux une fonction commune, c'est-à-dire la fonction de construction et de mémorisation d'un chemin survivant (CS). En effet, que ce soit pour la création d'un nouveau chemin survivant ou pour la mise à jour d'un chemin survivant existant, la fonction permettant d'ajouter des éléments à un chemin survivant est la même. Ainsi, avant d'aborder les étapes 2 et 3, il est jugé pertinent de présenter cette fonction.

La fonction de construction et de mémorisation d'un chemin survivant est récursive et s'exécute selon le pseudo-code de l'algorithme 1 où les différentes variables sont illustrées à la figure 8.16. En bref, cette fonction tente d'ajouter des éléments à un chemin survivant, et ce, tout en respectant les contraintes de déplacement illustrées à la figure 8.10. Pour sa part, la récursion permet de restreindre le nombre de tentatives d'enregistrement d'un chemin survivant et, par conséquent, permet d'accélérer considérablement le temps d'exécution de la fonction.

Selon la ligne 3 du pseudo-code de l'algorithme 1, pour être ajouté, l'élément de coordonnées (i,j) de la matrice de similarité (MS) doit correspondre à une valeur de similarité non nulle, tel que $MS(i,j) \neq 0$. Cette contrainte est logique puisqu'une valeur nulle impliquerait qu'aucune note ne partage la même identité, et ce, entre les notes contenues dans l'élément i de l'axe de la partition de musique et l'élément j de l'axe des notes détectées. Rappelons que le calcul des valeurs de similarité est présenté à la sous-section 8.2.1.

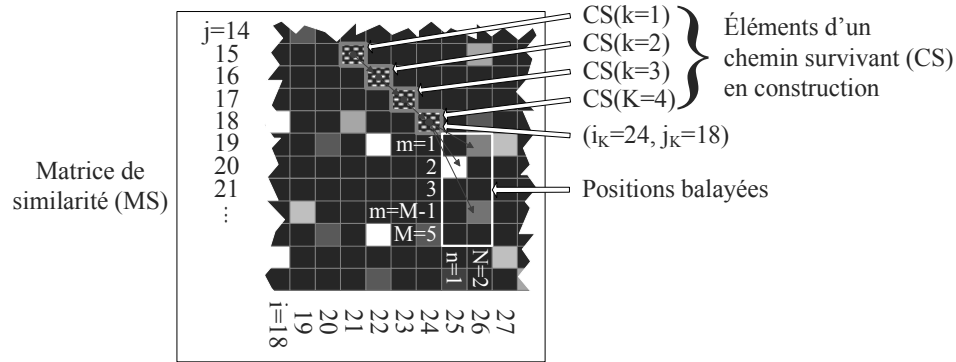


Figure 8.16 Illustration des variables utilisées par le pseudo-code de l'algorithme 1

```

Entrées :
CS(k) : Chemin survivant en construction composé de K éléments;
MS(i, j) : Matrice de similarité composée des lignes j et des colonnes i
Sorties :
Les chemins survivants complétés
Données :
i_K=Index de la colonne de la MS correspondant au dernier élément du CS;
j_K=Index de la ligne de la MS correspondant au dernier élément du CS;
/* Balayage des positions respectant les contraintes de déplacement */
1 pour m=1 à m=M faire
2   pour n=1 to n=N faire
3     si la valeur de similarité à MS(i_K + n, j_K + m) n'est pas nulle alors
4       ajouter l'élément (i_K + n, j_K + m) au CS;
5       si le CS résultant est plausible alors
6         /* Ici, la construction se poursuit par appels récursifs */
7         appeler cette fonction de nouveau avec le CS résultant comme entrée;
8         retirer tous les éléments ajoutés au CS lors des recursions précédentes
9       sinon
10        retirer l'élément ajouté au CS
11      fin
12    fin
13  fin
14 si la deuxième condition de cette fonction n'a jamais été valide alors
15   /* Ici, la composition actuelle du CS est maximale puisqu'il n'est plus possible
16     d'ajouter d'autres éléments. Par conséquent, ce n'est que maintenant qu'il
17     est tenté d'ajouter ce CS à la liste de tous les chemins survivants. */
18   si le CS respecte les conditions d'enregistrement alors
19     ajouter à la liste des chemins survivants
20   fin
21 fin

```

Algorithme 1 : Pseudo-code pour la construction de chemins survivants

En référence avec la ligne 5 du pseudo-code de l'algorithme 1, une fois l'élément ajouté, le CS résultant doit être plausible pour poursuivre la construction par appels récursifs. Pour être plausible, le CS doit respecter les trois conditions suivantes :

1. La valeur de confiance de l'élément ajouté calculée selon l'annexe C doit être suffisamment élevée.
2. Le ratio entre le nombre total d'éléments sautés pour l'axe de la partition (i) par rapport au nombre d'éléments dans le CS doit être suffisamment faible.
3. Les deux derniers éléments du CS en construction ne doivent pas chevaucher deux éléments d'un autre CS contenu dans la liste de CS retenus.

Il est à noter que pour les deux premières conditions, les valeurs seuils ont été ajustées à 0.8 et à 0.25 respectivement, et ce, dans le but d'obtenir de bons résultats. Pour sa part, la troisième condition permet d'arrêter rapidement les constructions qui convergent avec les éléments d'un CS existant. Ainsi, en évitant la redondance, cette condition permet d'accélérer le processus de construction de chemins survivants.

Selon la ligne 6 du pseudo-code de l'algorithme 1, une fois un élément ajouté au CS et les trois conditions précédentes respectées, la construction se poursuit par appels récursifs de cette même fonction. Cette construction peut résulter à la création de plusieurs chemins survivants, selon qu'une direction est prise par rapport à une autre. Ainsi, à partir de l'exemple de la figure 8.16, on pourrait imaginer quatre chemins survivants potentiels tels qu'illustrés à la figure 8.17, et ce, à condition que chacun des éléments ajoutés respecte les trois conditions précédentes.

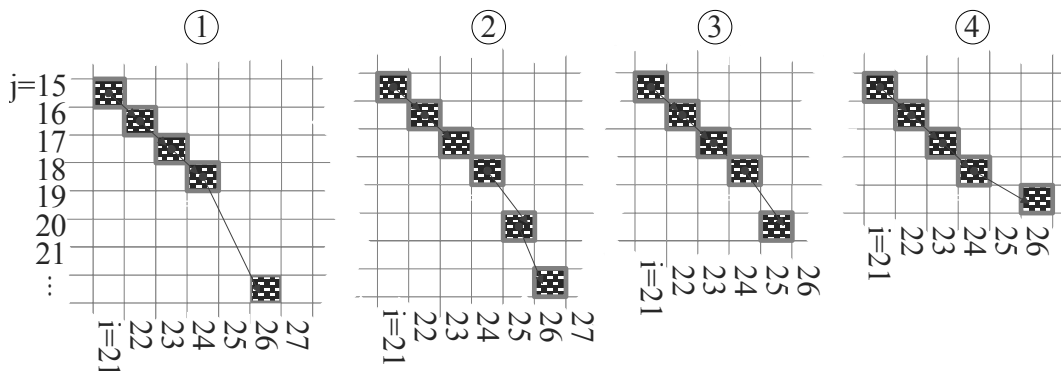


Figure 8.17 Exemple de chemins potentiels obtenus à partir de la figure 8.16

La ligne 15 du pseudo-code de l'algorithme 1 a pour objectif de restreindre la quantité de CS potentiels ajoutés à la liste de CS retenus, pour ce faire un CS doit satisfaire

certaines conditions pour être ajouté à la liste. En fait, ces conditions, énumérées ci-dessous, permettent essentiellement de contrôler la croissance de la liste de CS retenus, et ce, afin de permettre le fonctionnement en continu. Il est à noter que pour la première condition, le nombre minimal d'éléments est ajusté à 4, et ce, dans le but d'obtenir de bons résultats.

1. Le nombre d'éléments du CS à ajouter doit être supérieur ou égal à un nombre minimal.
2. Certains éléments du CS à ajouter doivent être différents du CS de la liste ayant le plus d'éléments communs.
3. Lorsque le CS à ajouter possède le même élément de départ et de fin qu'un CS de la liste, le CS sera ajouté seulement si son nombre total d'éléments sautés, pour l'axe de la partition (i) est inférieur au CS de la liste en question. En cas d'ajout, le CS de la liste concerné sera retiré, et ce, afin d'éviter la redondance.
4. Lorsqu'un CS de la liste possède tous les mêmes éléments que le CS à ajouter, mais que son nombre d'éléments est inférieur au CS à ajouter, le CS sera ajouté et le CS concerné de la liste sera retiré, et ce, afin de conserver le CS avec le plus d'éléments.

À titre d'exemple, imaginons que la liste de CS retenus soit vide et que le premier CS à ajouter soit le CS ① de la figure 8.17. Comme ce CS possède suffisamment d'éléments et qu'aucun CS n'est présent dans la liste, ce CS sera ajouté à la liste. Imaginons qu'il soit tenté ensuite d'ajouter le CS ②. Selon la quatrième contrainte, comme tous les éléments du CS ① sont communs à certains des éléments du CS ② à ajouter, le CS ② remplacera le CS ① contenu dans la liste.

Étape 2 : Mises à jour des chemins survivants existants

Selon la figure 8.13, une fois la mise à jour de la matrice terminée, la prochaine étape consiste à mettre à jour la liste des chemins survivants. D'abord, imaginons l'exemple fictif de la figure 8.18 contenant deux chemins survivants présentés à l'intérieur d'une matrice de similarité.

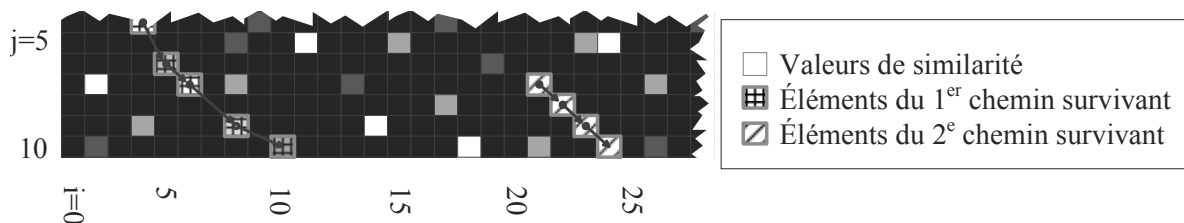


Figure 8.18 Exemple fictif de chemins survivants à l'intérieur d'une matrice de similarité

Imaginons maintenant que la matrice de similarité de la figure 8.18 soit mise à jour et que les valeurs de similarités obtenues soient celles de la figure 8.19. Pour ce cas, il est jugé important de signaler qu'il ne serait pas idéal de faire la mise à jour des deux chemins survivants en utilisant directement la fonction de construction d'un chemin survivant telle que décrite par le pseudo-code de l'algorithme 1 puisque, à partir du premier chemin survivant, les contraintes de déplacements telles qu'illustrées à la figure 8.10 ne permettraient pas d'obtenir le chemin survivant très crédible de la figure 8.20.

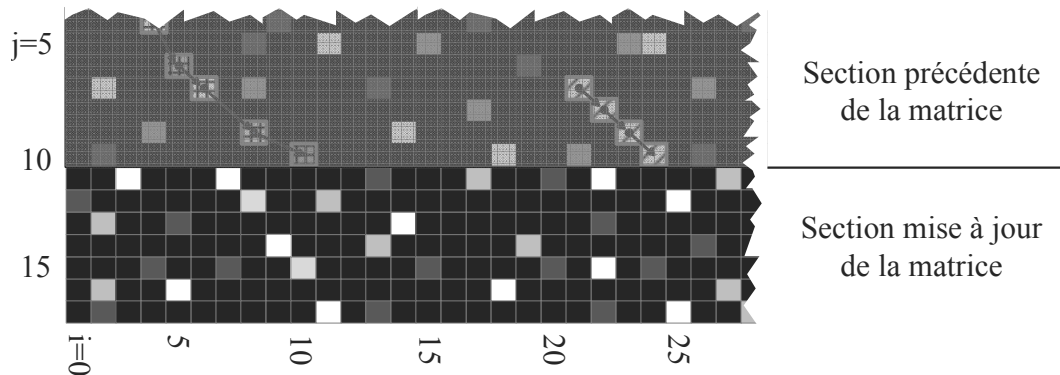


Figure 8.19 Matrice de la figure 8.18 à la suite d'une mise à jour

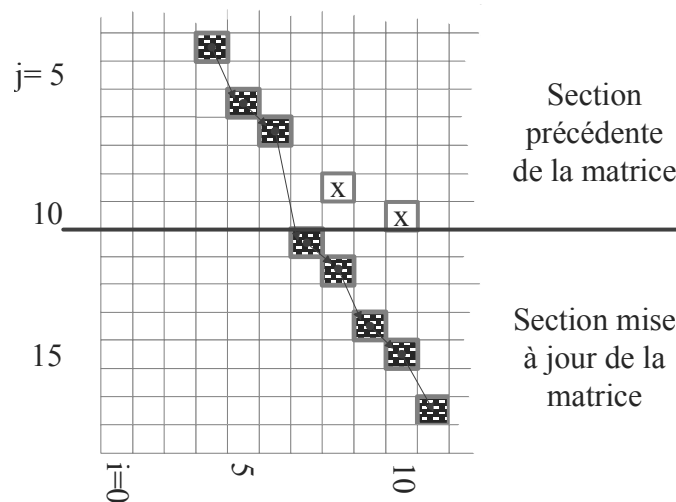


Figure 8.20 Chemin survivant très crédible à partir de la matrice de similarité de la figure 8.19

Ceci est dû au fait que la fonction de construction ajoute des éléments à partir du dernier élément du CS et que les contraintes de déplacements ne permettent pas les déplacements vers la gauche. Ainsi, pour obtenir le chemin survivant de la figure 8.20, il serait d'abord nécessaire de retirer les deux derniers éléments du chemin survivant avant d'appeler la fonction de construction, c'est-à-dire les deux éléments marqués d'une croix.

Par conséquent, la première étape à réaliser pour de la mise à jour des chemins survivants est de retirer le nombre adéquat d'éléments à la fin de chacun des chemins survivants, et ce, pour permettre une mise à jour complète. Ceci est effectué en retirant tous les éléments dont le numéro de ligne j est supérieur à la ligne j_{cut} . La ligne j_{cut} est trouvée à partir de l'équation 8.6 où M correspond au nombre de lignes contenues dans les contraintes de déplacements de la figure 8.10 et j_d correspond au numéro de la première ligne de la matrice de similarité mise à jour lors de la réalisation de l'étape 1. Il est à noter que pour l'exemple de la figure 8.19, la ligne j_d correspondrait à la ligne 11.

$$j_{cut} = j_d - M \quad (8.6)$$

Ainsi, une fois les éléments convenablement retirés, les chemins survivants peuvent être mis à jour en appelant, pour chacun d'eux, la fonction de construction et de mémorisation décrite par le pseudo-code de l'algorithme 1.

Étape 3 : Création de nouveaux chemins survivants

L'étape 3 de la figure 8.13 permet d'amorcer la construction de nouveaux chemins survivants. Elle est exécutée après l'étape de mise à jour des chemins survivants déjà existants, c'est-à-dire à la suite de l'étape 2. Cette séquence d'exécution donne priorité aux chemins survivants existants, et ce, pour éviter de construire un nouveau chemin survivant sur les traces d'un chemin survivant déjà existant. Ainsi, afin d'éviter la superposition redondante d'éléments, un nouveau chemin survivant doit toujours débiter sur un élément de la matrice de similarité sur lequel aucun autre chemin survivant n'a traversé.

Par conséquent, cette étape consiste d'abord à trouver une à une les valeurs de similarité non nulles traversées par aucun des chemins survivants et, ensuite, à appeler la fonction de construction et de mémorisation d'un chemin survivant décrite par le pseudo-code de l'algorithme 1 en utilisant cet élément comme entrée, et ce, sous la forme d'un CS composé d'un seul élément. Une fois appelée, la fonction se chargera de construire et de mémoriser les nouveaux chemins survivants débutant par cet élément. Il est à noter que pour obtenir de bons résultats, les colonnes de la matrice de similarité doivent être balayées une à une en démarrant par la première ligne mise à jour lors de l'étape 1, c'est-à-dire en démarrant par la ligne j_d définie à l'étape précédente.

Étape 4 : Fusion des chemins survivants

L'étape 4 de la figure 8.13 a pour objectif de fusionner des chemins survivants contigus. Afin d'illustrer cette étape, considérons les trois chemins survivants de la figure 8.21. À priori, on pourrait se demander pourquoi les chemins survivants 1 et 2 ne font pas partie

d'un seul chemin survivant. Ceci est dû au fait que la construction d'un chemin survivant peut être arrêtée lorsque l'une des trois conditions de la page 124 n'est pas respectée.

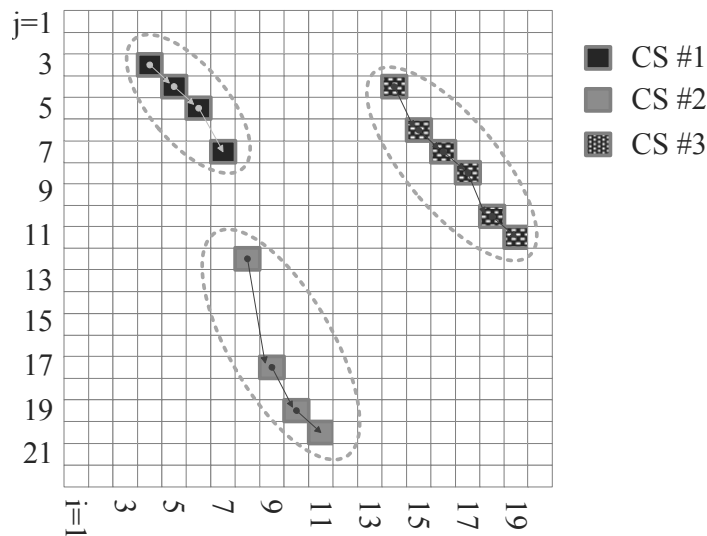


Figure 8.21 Exemple fictif de trois chemins survivants

Par exemple, on pourrait imaginer que lors de la construction du premier chemin survivant, la valeur de confiance obtenue en tentant d'ajouter l'élément de coordonnées $(i=8, j=12)$ n'était pas suffisamment élevée pour permettre cet ajout qui, pour la suite, s'avère être nécessaire à la poursuite de la construction. Rappelons que la valeur de confiance d'un élément d'un chemin survivant est obtenue par le biais des calculs présentés à l'annexe C. Ces calculs permettent essentiellement de vérifier les propriétés musicales suivantes pour chacun des éléments ajoutés à un chemin survivant :

1. Proximité de la position de début attendue par rapport à celle obtenue ;
2. Proximité des durées attendues par rapport à celles obtenues pour les notes contenues dans l'élément ajouté ;
3. Consistance du tempo impliqué par l'élément ajouté par rapport à celui des éléments précédents ;
4. Proximité du tempo prescrit dans la partition par rapport à celui obtenu pour l'élément ajouté ;
5. Intensités des notes potentielles contenues dans le chemin survivant par rapport à celles qui ont été sautées.

Par conséquent, une valeur de confiance faible implique que l'élément tenté d'être ajouté au chemin survivant n'est pas très compatible sur le plan musical. Ceci est généralement dû à l'une des situations suivantes :

1. Le chemin survivant en construction ne reflète pas vraiment ce qui est joué et, par conséquent, les éléments ajoutés ne sont pas compatibles sur le plan musical.
2. Le chemin survivant en construction reflète ce qui est joué, mais l'élément tenté d'être ajouté au chemin survivant n'est pas adéquat. Dans ce cas, le bon élément pourrait être ajouté lors d'une tentative subséquente.
3. Le chemin survivant en construction reflète ce qui est joué et l'élément tenté d'être ajouté au chemin survivant est le bon élément, mais le musicien n'a pas joué(s) correctement certaines des notes contenues dans l'élément produisant ainsi une faible valeur de confiance.
4. Le chemin survivant en construction reflète ce qui est joué et l'élément tenté d'être ajouté est le bon élément à ajouter, mais la couche de transcription automatique n'a pas correctement détecté(s) certaines des notes contenues dans l'élément.

Sachant ceci, imaginons que l'échec lors de l'ajout de l'élément de coordonnées $(i=8,j=12)$ du premier chemin survivant de la figure 8.21 était le résultat de la troisième ou de la quatrième situation énumérée ci-dessus et que l'élément de coordonnées $(i=9,j=17)$ était trop éloigné pour être ajouté à la suite de l'élément de coordonnées $(i=7,j=7)$, mais que si l'ajout de l'élément de coordonnées $(i=8,j=12)$ avait eu lieu, l'élément de coordonnées $(i=9,j=17)$ aurait été suffisamment proche pour être ajouté. Pour ce cas particulier, il est pertinent de fusionner les chemins survivants 1 et 3, comme en l'absence d'imperfections algorithmiques ou humaines, les chemins survivants 1 et 3 formeraient qu'un seul chemin survivant. Ceci est un exemple parmi d'autres illustrant la pertinence de cette étape de traitement, c'est-à-dire l'étape de fusion de chemins survivants.

Bien que, pour éviter cette étape de fusion de chemins survivants, il puisse sembler préférable de simplement diminuer le seuil de la valeur de confiance minimale nécessaire à l'acceptation de l'ajout d'un élément à un chemin survivant, dans les faits, ceci conduirait à de faibles performances. Ceci est dû au fait que, pour obtenir de bonnes performances, tous les éléments d'un chemin survivant doivent être solides sur le point de vue musical. Ainsi, malgré qu'un chemin survivant de plusieurs éléments ne corresponde pas au segment de la partition joué, ce chemin survivant reflétera nécessairement un autre segment de la partition ayant une grande ressemblance sur le point de vue musical.

Pour effectuer le fusionnement de chemins survivants, un bloc d'analyse de dimensions $N \times M$ doit d'abord être positionné sur le dernier élément de chacun des chemins survivants enregistrés à l'étape 3 de la figure 8.13 tel qu'illustré par l'exemple fictif de la figure 8.22 pour un bloc de dimensions 4×4 . Une fois positionnés, tous les chemins survi-

vants passant à l'intérieur d'un bloc d'analyse d'un autre chemin survivant sont automatiquement fusionnés avec ce dernier. Il est à noter que pour l'évaluation de la performance du chapitre 12, un bloc d'analyse de dimensions 7×7 a été utilisé.

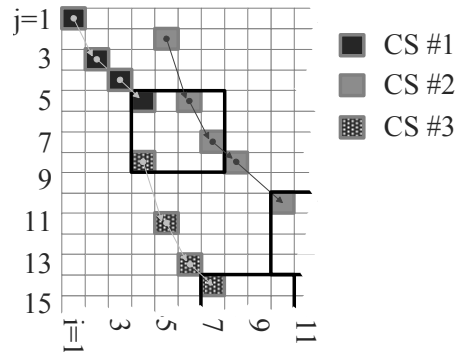


Figure 8.22 Exemple de positionnement de blocs d'analyse de dimensions 4×4

Lors du fusionnement, les diverses combinaisons possibles sont vérifiées une à une, et ce, avec un maximum d'un élément par colonne. Par exemple, pour la figure 8.22, les diverses combinaisons possibles sont illustrées à la figure 8.23 où les fusions ① et ② sont le résultat de la fusion entre le premier et le troisième chemin survivant et les fusions ③ et ④ sont le résultat de la fusion entre le premier et le deuxième chemin survivant.

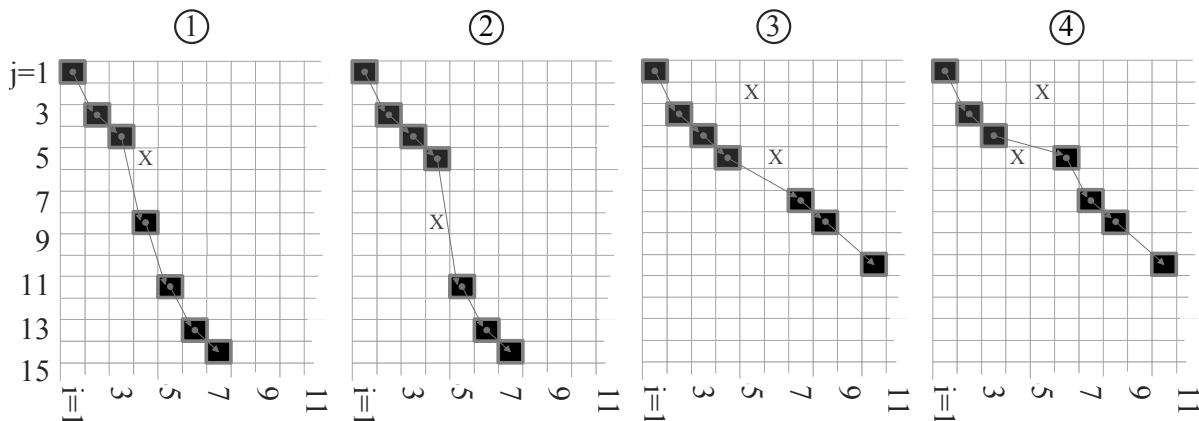


Figure 8.23 Combinaisons possibles de fusionnement à partir des chemins survivants de la figure 8.22

Afin de limiter le nombre de chemins survivants fusionnés à retenir, et ce, dans le but de favoriser le fonctionnement en continu, un seul chemin survivant par fusion de chemins survivants de même identité est ajouté à la liste de chemins survivants fusionnés. Par exemple, comme les chemins survivants ① et ② correspondent à la fusion de deux chemins survivants de même identité, un seul d'entre eux sera tenté d'être ajouté à la liste.

Afin de sélectionner le chemin survivant à retenir, les valeurs de confiance sont d'abord calculées pour chacun des éléments des chemins survivants fusionnés, et ce, toujours selon l'annexe C. Une fois les valeurs calculées, le chemin survivant que l'on tentera d'ajouter à la liste sera celui ayant la plus haute moyenne de ses valeurs de confiance. Ainsi, toujours dans le but de limiter le nombre de chemins survivants à retenir, pour être ajouté à la liste, un chemin survivant doit respecter les mêmes conditions que celles énumérées à la page 125. Il est à noter que pour accélérer le temps d'exécution, une mécanique récursive similaire à celle du pseudo-code l'algorithme 1 est appliquée pour réaliser cette étape de traitement. La principale différence est qu'au lieu d'ajouter des éléments à un chemin survivant, des chemins survivants y sont fusionnés.

Étape 5 : Sélection des chemins survivants fusionnés pour réaliser l'alignement

Pour le cas particulier où un musicien jouerait une partition de musique du début jusqu'à la fin, la liste de chemins survivants contiendrait normalement un chemin survivant couvrant la majorité de l'axe de la partition de musique et d'autres petits chemins survivants tel qu'illustré par l'exemple fictif de la figure 8.24. Pour cet exemple, il apparaît évident que l'alignement devrait correspondre au chemin survivant le plus long, c'est-à-dire le chemin survivant 1. Par contre, lorsque le musicien décide de sauter d'un endroit à l'autre dans la partition de musique, l'alignement peut-être composé de plusieurs chemins survivants tel qu'illustré à la figure 8.25 où le musicien a commencé à jouer la partition du milieu jusqu'à la fin et a ensuite joué du début jusqu'au milieu.

Cela dit, l'objectif de cette étape de traitement consiste à sélectionner les meilleurs candidats parmi les chemins survivants de l'étape 4 dans le but de réaliser l'alignement brut, et ce, de façon à donner la liberté au musicien de jouer les segments de la partition qu'il désire sans préalablement avoir à informer le système de ses intentions.

Tout comme pour l'étape de traitement 4, cette étape est réalisée en fusionnant des chemins survivants, mais cette fois avec des contraintes de fusionnement plus flexibles relativement à l'axe de la partition de musique. Pour ce faire, au lieu d'utiliser un bloc d'analyse, trois limites horizontales (A , B et C) sont positionnées à partir du dernier élément de chacun des chemins survivants telles qu'illustrées à la figure 8.26 où, par rapport au dernier élément, la limite A est positionnée deux lignes avant, la limite B est positionnée deux lignes

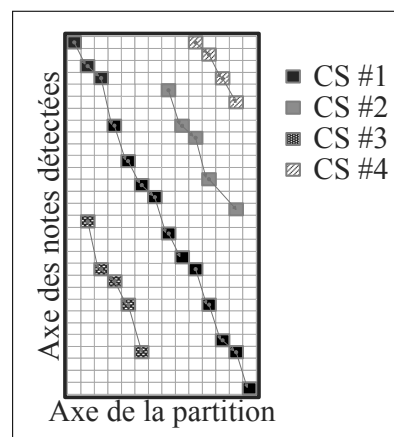


Figure 8.24 Exemple fictif de chemins survivants obtenus lorsque le musicien joue du début jusqu'à la fin une partition de musique

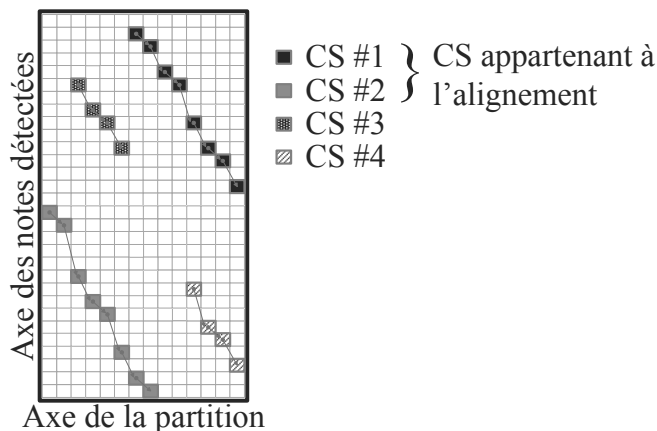


Figure 8.25 Exemple fictif de chemins survivants obtenus lorsque le musicien joue la partition de musique à partir du milieu jusqu'à la fin et ensuite du début jusqu'au milieu

après et la limite C est positionné trois lignes après. Il est à noter que ces limites ont été positionnées afin d'obtenir de bons résultats.

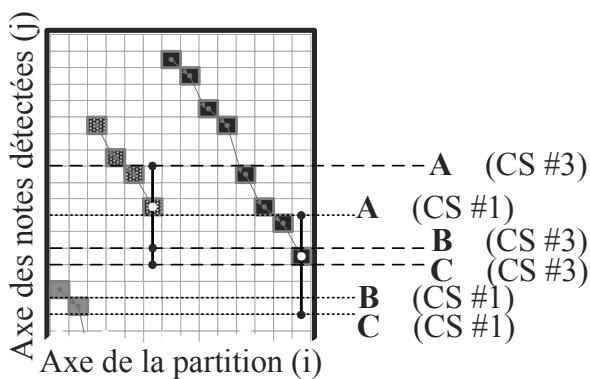


Figure 8.26 Positionnement des limites de fusionnement

Ainsi, pour être fusionné à un autre chemin survivant, un chemin survivant doit respecter les conditions suivantes :

1. Le numéro de ligne de son premier élément doit être à l'intérieur des limites A et B .
2. Le numéro de ligne de son dernier élément doit être supérieur à la limite C .

Pour le cas où aucun chemin survivant ne respecterait ces deux contraintes, et ce, pour un chemin survivant donné, la première contrainte serait exclue dans le but d'obtenir une combinaison de chemins survivants couvrant la majorité de l'axe j des notes détectées. Cette technique de fusionnement permet d'obtenir plusieurs combinaisons de chemins survivants où chacune de ces combinaisons correspond à un alignement brut potentiel.

Comme plusieurs combinaisons risquent d'être trouvées, il s'avère nécessaire de pondérer chacune de ces possibilités. Ainsi, la sortie du bloc d'alignement brut est composée de plusieurs alignements bruts accompagnés d'une valeur de confiance. Cette valeur de confiance $c_a[l]$ associée à un alignement brut d'indice l est calculée à l'aide des équations 8.7, 8.8, 8.9 et 8.10 où $j[l,n]$ correspond au numéro de ligne de la matrice de similarité correspondant au n^{ieme} élément de la combinaison de N éléments, la valeur $\bar{c}[l]$ correspond à la moyenne de toutes les N valeurs de confiance comprises à l'intérieur de tous les chemins survivants faisant parti de la combinaison et, finalement, les valeurs j_{min} et j_{max} correspondent à la ligne minimale et maximale de la matrice de similarité respectivement. Rappelons que la matrice de similarité oublie les lignes au-delà d'un délai tel qu'illustré à la figure 8.14. Il est à noter que pour l'évaluation de la performance du chapitre 12, la constante cst de l'équation 8.8 a été fixée à 5.

$$\gamma[l] = \min \left(1, \frac{1 + j[l, N] - \max(j[l, 1], j_{min})}{1 + j_{max} - j_{min}} \right) \quad (8.7)$$

$$\eta[l] = \sum_{n \in j[l,n] | (j[l,n+1] - j[l,n] - 1) \geq cst} (j[l, n + 1] - j[l, n] - 1) \quad (8.8)$$

$$\beta[l] = \frac{N}{N + \eta[l]} \quad (8.9)$$

$$c_a[l] = \bar{c}[l] \gamma[l] \beta[l] \quad (8.10)$$

Afin d'aider à la compréhension, l'exemple de la figure 8.27 présente le calcul de la valeur de confiance de l'alignement brut composé des deux chemins survivants, et ce, en admettant que la valeur de $\bar{c}[l]$ de l'équation 8.10 soit de 0.87 et que la valeur de cst de l'équation 8.8 soit de 5.

En quelques mots, la variable $\gamma[l]$ a pour objectif de s'assurer d'aligner toute la plage de notes jouées par le musicien. Quant à elle, la variable $\beta[l]$ a pour objectif d'assurer que la combinaison de chemins survivants laisse le moins de vide possible, c'est-à-dire des instants dans le temps où l'on ne sait pas trop ce que fait le musicien.

Pour terminer, cette étape de traitement permet de trouver plusieurs alignements bruts potentiels qui seront utilisés ultérieurement par le bloc subséquent d'alignement fin. Suite à une analyse exhaustive, le bloc d'alignement fin sélectionnera parmi tous ces candidats

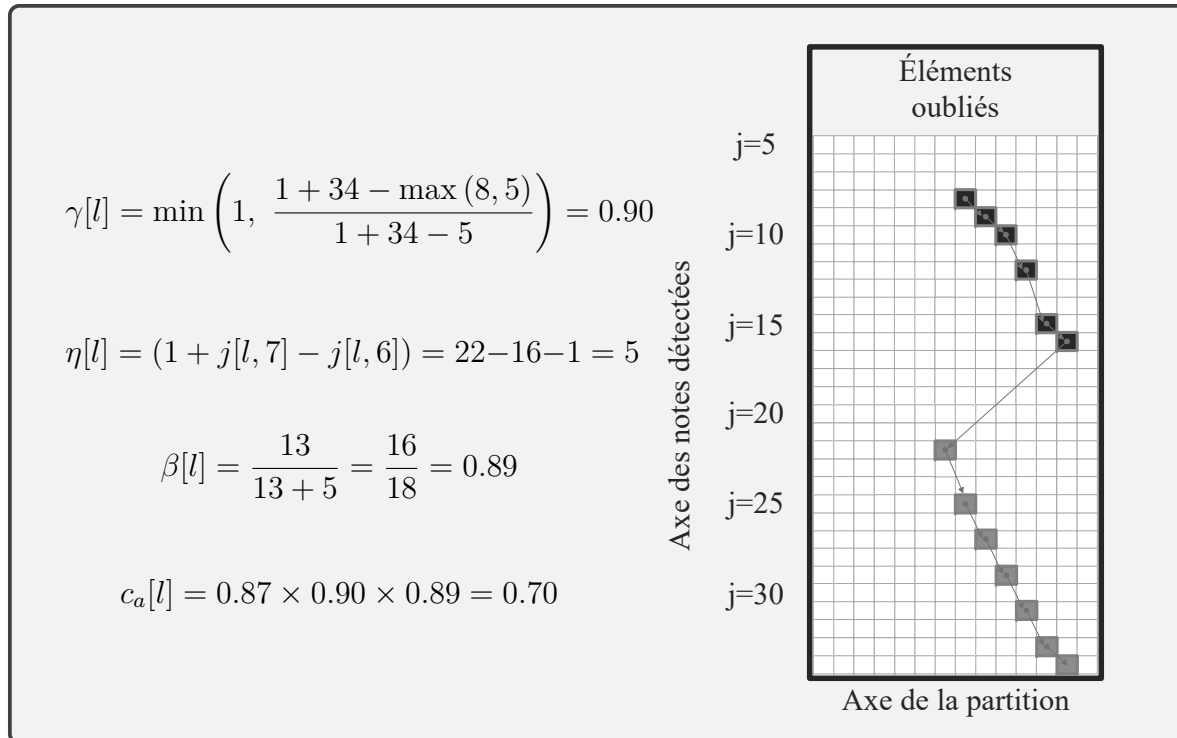


Figure 8.27 Exemple de calcul de la valeur de confiance d'un alignement brut

celui qu'il jugera le plus probable, et ce, en commençant par analyser celui ayant la plus grande valeur de confiance. Pour assurer le fonctionnement en continu, la couche d'alignement fin disposera d'un temps limite, une fois le délai échu, certains alignements bruts peuvent ne pas être vérifiés d'où l'importance de leur associer une valeur de confiance.

Étape 6 : Maintenance de la liste de chemins survivants

Lors du premier passage à travers les six étapes de traitement de la figure 8.13, l'étape 2 de mises à jour des chemins survivants ne peut pas se produire comme la liste de chemins survivants est vide à ce point. En fait, la liste n'est démarrée qu'à partir de l'étape 3, c'est-à-dire l'étape de création des chemins survivants. Cette liste est ensuite modifiée par l'étape 4 pour devenir la liste des chemins survivants fusionnés. Pour terminer, la liste de l'étape 4 est récupérée de nouveau par l'étape 2, et ce, une fois le passage à travers les diverses étapes de la figure 8.13 effectué. Sans entretien, la dimension de cette liste augmenterait continuellement de passage en passage. À long terme, cette dimension deviendrait trop importante pour permettre le fonctionnement de l'algorithme. Afin d'éviter ce problème, l'étape de traitement 6 a pour objectif de contrôler la croissance de cette liste. En bref, les moyens de contrôle de croissance utilisés sont les suivants :

1. **Retrait des CS expirés :** Retrait des chemins survivants se terminant à une ligne inférieure à la première ligne conservée de la matrice de similarité ;

2. **Retrait des CS moins probables** : Retrait des chemins survivants les plus courts en commençant par les plus vieux, c'est-à-dire ceux ayant le moins d'éléments et se terminant par un numéro de ligne inférieur à tous les autres de même dimension ;
3. **Retrait des CS similaires** : Retrait des chemins survivants dont plusieurs éléments sont communs à un autre CS de dimension supérieure.

Contrairement au premier moyen de contrôle, les deux autres ne sont pas effectués à chacun des passages à travers l'étape de traitement 6. En fait, le deuxième moyen de contrôle est effectué seulement si la dimension de la liste excède une valeur prédéterminée ajustée à 40 pour les résultats du chapitre 12. En cas de dépassement, la liste est réduite via le deuxième moyen de contrôle, et ce, jusqu'à ce qu'elle possède une dimension inférieure à cette valeur prédéterminée. Pour sa part, le troisième moyen de contrôle est seulement effectué une fois tous les dix passages. Ce choix s'explique par le fait que ce moyen de contrôle est très long à exécuter comme il exige un nombre considérable de comparaisons au niveau des éléments de chacun des chemins survivants. Pour ce dernier, le ratio entre le nombre d'éléments communs et le nombre total d'éléments d'un chemin survivant à retirer doit être supérieur à une valeur ajustée à 0.85, et ce, dans le but d'obtenir de bons résultats.

Il est à noter que le deuxième moyen de contrôle de croissance peut s'avérer très dommageable puisqu'il peut retirer par erreur des chemins survivants utiles à l'alignement. Pour éviter ces retraits problématiques, plusieurs aires de la matrice de similarité sont protégées. Ainsi, un chemin survivant passant à travers une aire de protection ne pourra pas être retiré par ce moyen de contrôle. Ces aires de protection sont déterminées par les alignements bruts obtenus lors de l'étape de traitement précédente. Les quatre extrémités (i_A, j_A) , (i_B, j_A) , (i_A, j_B) et (i_B, j_B) d'une aire de protection telles qu'illustrés à la figure 8.28 pour un alignement brut d'indice l sont ajustées à l'aide des équations 8.11, 8.12, 8.13 et 8.14 où $j[l, 1]$ et $i[l, 1]$ correspondent au numéro de ligne et de colonne respectivement associé au premier élément de l'alignement brut. Pour leur part, les variables j_{min} et j_{max} correspondent respectivement à la première et dernière ligne conservée de la matrice de similarité et i_{max} au nombre de colonnes dans la matrice de similarité. Il est à noter que pour les résultats du chapitre 12, la constante cst de l'équation 8.13 a été ajustée à 10.

Selon la figure 8.13, cette dernière étape met fin à la série de six étapes de traitement répétées en boucle.

$$j_A = \max(j_{min}, j[l, 1]) \quad (8.11)$$

$$j_B = j_{max} \quad (8.12)$$

$$i_A = \max(1, i[l, 1] - cst) \quad (8.13)$$

$$i_B = \min(i_{max}, j[l, 1] + (j_B - j_A)) \quad (8.14)$$

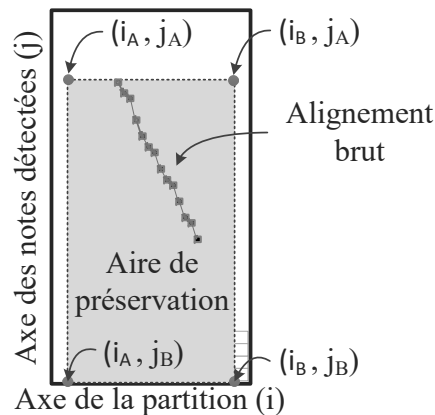


Figure 8.28 Représentation d'une aire protégée d'une matrice de similarité

8.2.3 Délai d'expiration

Comme on peut le remarquer, plusieurs attentions particulières ont été accordées aux diverses techniques de fonctionnement telles que la technique de construction par récursion d'un chemin survivant et la technique de contrôle de croissance d'une liste de chemins survivants. Ces détails parmi d'autres sont d'une grande importance pour assurer un fonctionnement en temps réel de type meilleur effort. Néanmoins, malgré toutes ces précautions, il arrive parfois que le temps requis pour exécuter certaines étapes de traitement soit trop grand pour assurer le fonctionnement en temps réel. Afin de pallier ce problème, des comptes à rebours sont appliqués aux étapes de traitement 2, 3 et 4. Ainsi, chacune de ces étapes de traitement dispose d'un certain temps pour exécuter leur travail. Une fois le délai d'une étape échu, seuls les chemins survivants ayant eu le temps d'être enregistrés lors de cette étape peuvent être transmis à l'étape subséquente.

Il est à noter que les comptes à rebours sont généralement utiles en présence de segments où plusieurs notes de même identité sont répétées continuellement tels que pour l'extrait de la figure 8.29 provenant du deuxième mouvement de la deuxième sonate pour violon de Prokofiev [59].



Figure 8.29 Exemple de segment de partition répétitif

D'ailleurs, la figure 8.30 illustre la matrice de similarité obtenue pour un extrait de cette sonate où la partie encerclée correspond au segment répétitif de la figure 8.29. Pour cet

exemple, la construction de chemins survivants à travers ce bloc répétitif peut prendre un temps de calcul non négligeable puisque, comme plusieurs directions sont possibles, chacun des chemins survivants en construction peut se dupliquer pour créer d'autres chemins survivants de directions différentes pouvant à leur tour se dupliquer et ainsi de suite. Il est à noter que cette prolifération s'explique par le fonctionnement interne de la fonction de construction définie par le pseudo-code de l'algorithme 1.

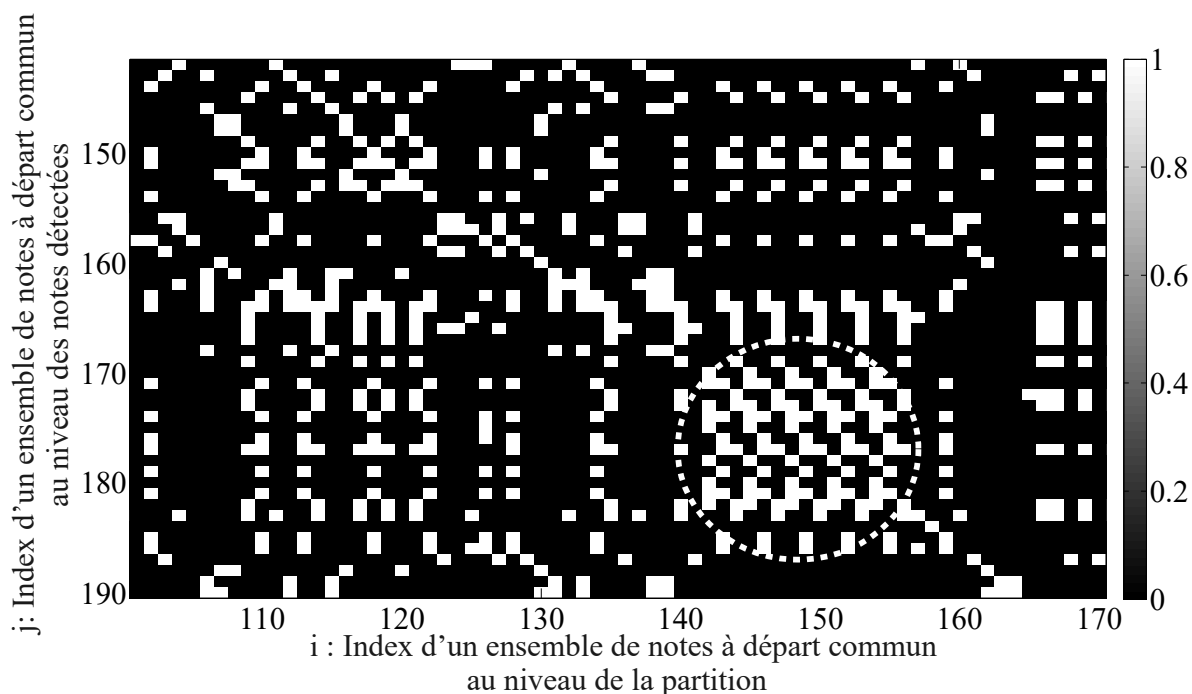


Figure 8.30 Extrait d'une matrice de similarité en présence d'un segment répétitif

8.2.4 Justification de l'algorithme EDTW

L'algorithme EDTW présenté dans cette section a été conçu suite à l'étude des diverses méthodes d'alignement retrouvées dans la littérature scientifique, et ce, en lien avec le contexte visé par ce projet de recherche. Tel que discuté au chapitre 3, ces diverses méthodes sont essentiellement basées sur les algorithmes DTW [2, 12, 28, 69, 70, 73, 76] et HMM [8, 51, 54, 63, 72] ainsi que sur les modèles bayésiens [43, 64, 65]. Conséquemment, l'algorithme EDTW a été conçu pour converger dans un même algorithme les principaux avantages de ces trois algorithmes et, aussi, pour intégrer de nouveaux avantages.

D'abord, comme son nom l'indique, l'algorithme DTW tolère les variations dans le temps. Ainsi, les méthodes d'alignement basées sur l'algorithme DTW fonctionnent correctement lorsqu'une pièce de musique à jouer est interprétée avec un tempo différent de celui attendu et interprétée avec diverses variations de tempo telles que lors d'accélération (*accelerando*)

et de décélérations (*diminuendo*). En contrepartie, les méthodes basées sur l'algorithme HMM sont entraînées pour s'adapter aux diverses interprétations d'un musicien. Ainsi, advenant qu'un musicien joue une pièce avec le même tempo et les mêmes variations de tempo d'une répétition à l'autre pendant les apprentissages, la méthode aura de la difficulté à aligner correctement une interprétation dont le tempo est beaucoup plus lent ou rapide et dont les variations de tempo sont différentes de celui et de celles encourus lors des entraînements. Cette capacité d'apprentissage assure davantage de robustesse en présence d'interprétations similaires à celles utilisées pendant l'entraînement. Tout comme pour les méthodes basées sur l'algorithme HMM, les méthodes basées sur les modèles bayésiens permettent aussi l'entraînement, mais supportent mieux un changement de tempo, et ce, comme la méthode utilise des variables aléatoires non observables qui s'ajustent en fonction du tempo local.

Ceci dit, tout comme pour les méthodes basées sur l'algorithme DTW et sur les modèles bayésiens, l'algorithme EDTW a été conçu pour supporter les changements et les variations de tempo. En effet, l'algorithme EDTW évalue un tempo local en fonction des éléments retrouvés dans chacun des chemins survivants analysés, et ce, tout au long de l'alignement. Par contre, l'algorithme EDTW ne permet pas l'entraînement, et ce, comme l'objectif visé était de créer un algorithme générique qui ne nécessiterait pas d'apprentissage. D'ailleurs, un professeur de musique n'a pas besoin que le musicien joue plusieurs fois la même pièce de musique pour être en mesure d'établir la correspondance entre les notes jouées et celles de la partition.

D'autant plus, l'algorithme EDTW a été conçu pour fonctionner en présence de pièces polyphoniques, et ce, comparativement à l'algorithme DTW. Aussi, en comparaison avec tous les autres algorithmes, l'algorithme EDTW donne la flexibilité au musicien de jouer librement les différents segments d'une partition de musique, et ce, tel que discuté précédemment. De plus, en traitant des notes au lieu de des trames, l'algorithme EDTW a l'avantage d'être plus léger au niveau de l'utilisation CPU et mémoire que tous les autres algorithmes. Finalement, tout comme pour le DTW, la représentation des divers alignements estimés et des divers chemins survivants à l'intérieur d'une matrice de similarité facilite le débogage.

En contrepartie, l'algorithme EDTW possède un temps de latence légèrement supérieur aux autres algorithmes, et ce, comme les notes estimées doivent d'abord être complétées par la couche de transcription automatique, et ce, avant de pouvoir être traitées par la couche d'alignement. Par conséquent, pour pouvoir suivre un musicien avec un temps de latence équivalent aux autres algorithmes, il faudrait réaliser une nouvelle version du

système qui permettrait à la couche de transcription automatique de transmettre des notes en construction, et ce, tout en spécifiant leur nature incomplète.

La prochaine section présente le deuxième bloc d'alignement, c'est-à-dire le bloc d'alignement fin de la figure 8.2.

8.3 Alignement fin

En admettant que la vitesse et la mémoire des ordinateurs personnels d'aujourd'hui soient illimitées, la meilleure façon d'obtenir l'alignement optimal serait probablement d'essayer toutes les combinaisons possibles d'agencement de notes potentielles provenant de la couche de transcription automatique du chapitre 7 avec les notes provenant de la partition de musique. Ensuite, de choisir celle donnant la meilleure moyenne des valeurs de confiance calculées selon des calculs similaires aux sections C.2 à C.8 de l'annexe C, c'est-à-dire des calculs permettant de vérifier la compatibilité des notes sur le plan musical.

À titre d'exemple, imaginons les notes du tableau 8.4 représentant une partition de musique simpliste de quatre notes. En admettant que le musicien ne soit pas parfait, il pourrait oublier de jouer certaines de ces notes ou même jouer des notes en trop. Par conséquent, toutes les possibilités de combinaisons de notes de la partition pouvant être jouées, en admettant que le musicien ne joue pas la partition de musique à reculons, peuvent être représentées graphiquement par le biais d'un arbre tel qu'illustré à la figure 8.31. Par exemple, le trajet représenté par les flèches plus denses représenterait le cas où le musicien aurait oublié de jouer la note Ré. Il est à noter que le nombre de combinaisons possibles pour une partition de musique composée de N notes monophoniques est de 2^N , soit une croissance exponentielle.

Tableau 8.4 Exemple de notes représentant une partition de musique

Notes	Do4	Ré4	Mi4	Fa4
Début (mesures)	1.0	1.25	1.50	1.75
Fin (mesures)	1.25	1.50	1.75	2.0

Admettons maintenant que les notes potentielles détectées par la couche de transcription automatique soient celles du tableau 8.5. À partir de ces nouvelles données, il est possible d'ajuster l'arbre de la figure 8.31 pour maintenant obtenir l'arbre de la figure 8.32 où chacune des notes de la partition est liée à une note potentielle de même identité.

Il est à noter que cet arbre tient compte du fait que certaines notes potentielles pourraient être des notes détectées en trop, c'est-à-dire des notes que le musicien n'a pas réellement jouées mais, que la couche de transcription automatique a tout de même trouvées par

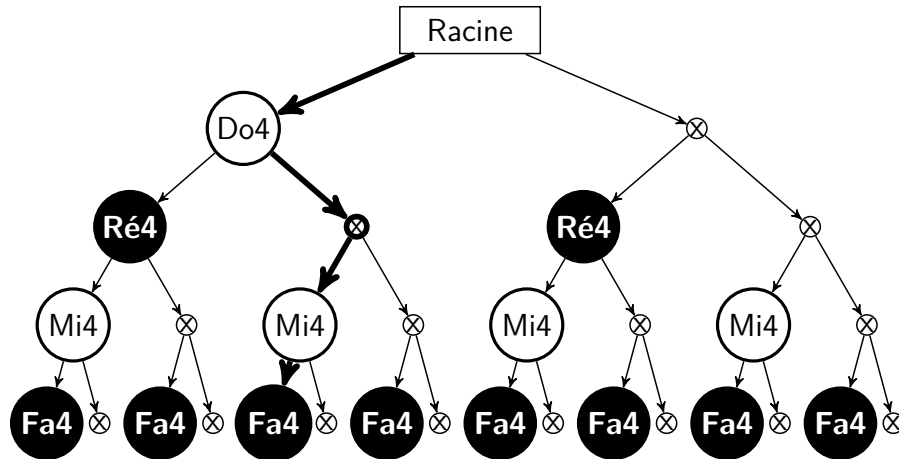


Figure 8.31 Représentation en arbre des combinaisons possibles de notes de la partition

Tableau 8.5 Exemple de notes potentielles détectées

No.	①	②	③	④	⑤	⑥
Notes	Ré4	Do4	Ré4	Mi4	Fa4	Do4
Début (s.)	0.5	1.5	2.1	2.6	3.2	3.7
Fin (s.)	1.5	2.1	2.6	3.2	3.7	4.1

erreur. En effet, bien qu'à priori il puisse sembler évident que les notes de la partition devraient correspondre aux notes potentielles ② à ⑤ du tableau 8.4, il se peut dans les faits qu'une ou plusieurs de ces notes potentielles n'aient pas réellement été jouées. D'ailleurs, dans un tel cas, les notes détectées en trop seraient probablement incompatibles sur le plan musical et, par conséquent, donneraient une valeur de confiance faible, et ce, à partir de calculs similaires à ceux utilisés pour évaluer les valeurs de confiance des éléments d'un chemin survivant du bloc d'alignement brut tels que détaillés aux sections C.2 à C.8 de l'annexe C.

À partir de la figure 8.32, on peut remarquer que le nombre total d'alignements possibles est de 21, c'est-à-dire la somme de tous les éléments du dernier niveau. Étant donné la croissance exponentielle du nombre d'alignements possible pour un nombre donné de notes de la partition et de notes potentielles, tenter de vérifier tous les alignements possibles d'une partition de musique composée de plusieurs dizaines de notes pour obtenir l'alignement optimal risquerait de prendre un temps de traitement trop important pour assurer un fonctionnement en temps réel de type meilleur effort. Malgré tout, l'alignement fin construit un arbre similaire à celui de la figure 8.32, mais pour sauver du temps de traitement, l'algorithme d'alignement fin ne développe pas toutes les branches de l'arbre. En

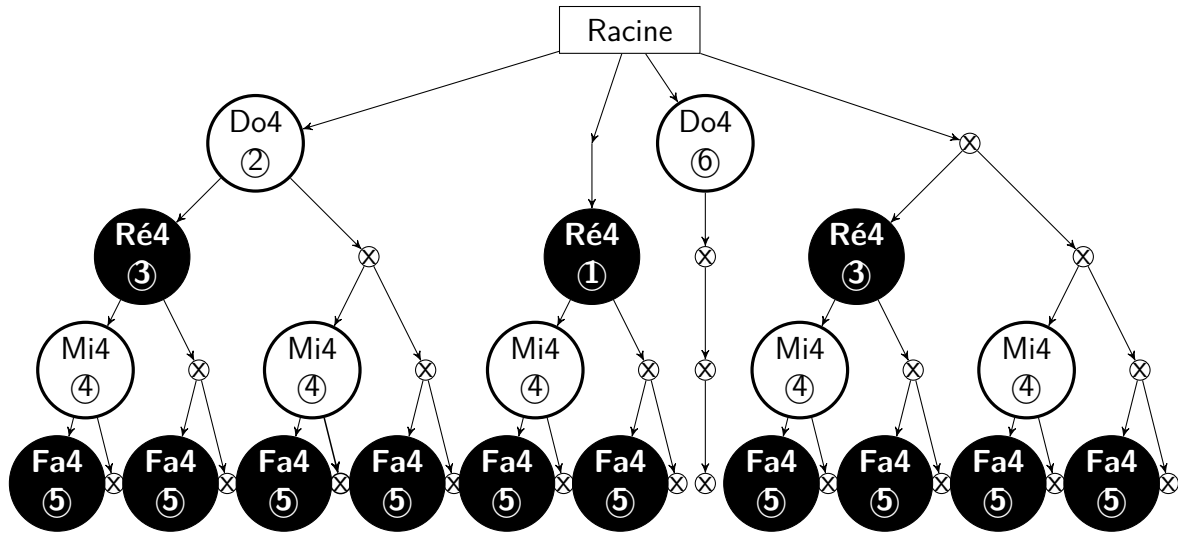


Figure 8.32 Représentation en arbre de toutes les possibilités

fait, l'algorithme utilise, entre autres, l'alignement brut pour circonscrire le développement de l'arbre.

Afin d'expliquer ceci par le biais d'un exemple, imaginons que les notes Do, Ré et Fa du tableau 8.4 soient alignées à l'aide du bloc d'alignement brut aux notes ②, ③ et ⑤ du tableau 8.5. D'abord, malgré que l'alignement brut n'ait pas réussi à aligner la note Mi de la partition de musique, son travail permet tout de même d'approximer la position attendue de la note Mi dans le temps. Cette approximation est faite selon l'équation 8.15 où la variable m correspond à la position de début en mesure de note dont le début s en seconde est approximé et les variables s_g , s_d , m_g et m_d correspondent à des positions de début de notes faisant parties de l'alignement brut. Plus précisément, par rapport à la note dont le début est approximé, s_g et s_d correspondent respectivement au début en secondes de la note à gauche et à droite et m_g et m_d correspondent respectivement au début en mesure de la note à gauche et à droite.

$$s = s_g + \left(\frac{s_d - s_g}{m_d - m_g} \right) (m - m_g) \quad (8.15)$$

Par conséquent, pour l'exemple précédent, la position attendue de la note Mi en secondes serait trouvée comme suit, et ce, en utilisant les données des tableaux 8.4 et 8.5 :

$$s = 2.1 + \left(\frac{3.2 - 2.1}{1.25 - 1.75} \right) (1.5 - 1.25) = 2.65$$

Une fois que toutes les positions de début des notes manquantes sont approximées, le bloc d'alignement fin définit des plages temporelles où la position de début de chacune des notes de la partition de musique est attendue telles qu'illustrées à la figure 8.33, et pour fin pratique, résumées dans le tableau 8.6.

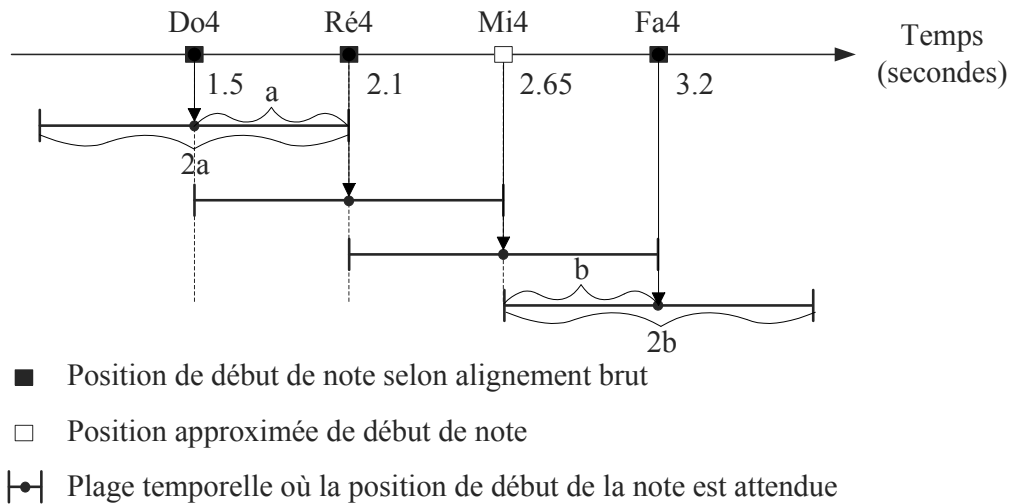


Figure 8.33 Plages temporelles où la position de début de chacune des notes de la partition de musique est attendue

Tableau 8.6 Plages temporelles où la position de début de chacune des notes de la partition de musique est attendue

Notes	Do4	Ré4	Mi4	Fa4
Plage temporelle (s.)	[0.9, 2.1]	[1.5, 2.65]	[2.1, 3.2]	[2.65, 3.75]

Ces plages temporelles permettent de réduire le nombre d'alignements possibles. Par exemple, en retirant toutes les branches de l'arbre de la figure 8.32 où l'un des éléments ne respecte pas les plages définies du tableau 8.6, l'arbre de la figure 8.34 serait obtenu, et ce, toujours en supposant que les notes ne seront jamais jouées à reculons. Néanmoins, malgré cette réduction, la dimension de l'arbre demeure tout de même importante. C'est pourquoi le bloc d'alignement fin utilise une autre méthode pour réduire davantage la dimension de l'arbre. Cette méthode consiste à construire seulement les N meilleures branches de l'arbre, et ce, à chacun des étages de l'arbre. Les meilleures branches choisies sont celles dont le nombre de notes manquantes \otimes est le plus petit. Pour les résultats du chapitre 12, la valeur de N a été ajustée à 20, et ce, dans le but d'assurer un fonctionnement en temps réel de type meilleur effort tout en obtenant le maximum de performance.

Par exemple, en admettant que le nombre N soit ajusté à 4, l'arbre de la figure 8.34 deviendrait l'arbre de la figure 8.35. De cet exemple, on remarque que déjà au troisième

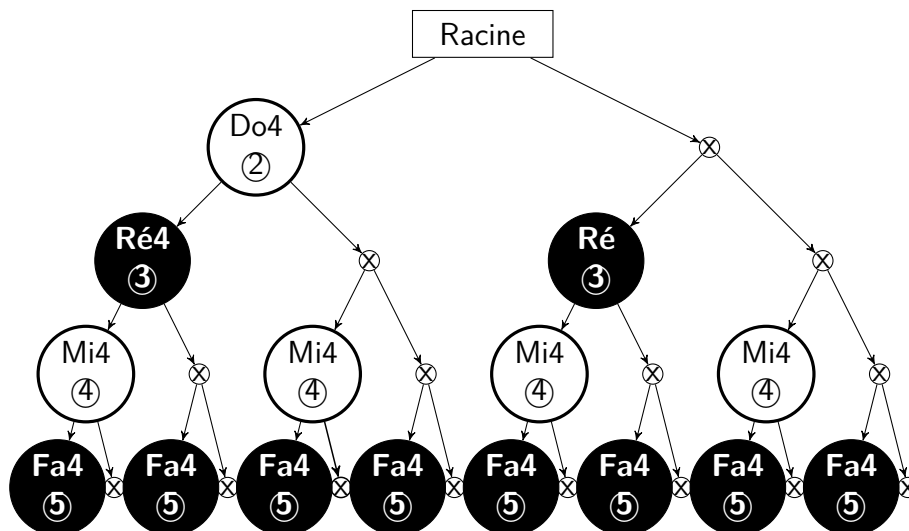


Figure 8.34 Représentation en arbre simplifié selon les contraintes temporelles

niveau, seulement quatre des huit branches de la figure 8.34 ont été construites. Néanmoins, un problème se pose au quatrième niveau, car cinq branches sont conservées au lieu de quatre. Ceci est dû au fait que le quatrième niveau possède quatre branches ayant un nombre égal de notes manquantes, c'est-à-dire une note manquante sur quatre notes. Afin de décider laquelle de ces quatre branches doit être éliminée, la compatibilité sur le plan musical est trouvée selon des calculs similaires à ceux utilisés pour évaluer les valeurs de confiance des éléments d'un chemin survivant du bloc d'alignement brut tels que détaillés aux sections C.2 à C.8 de l'annexe C.

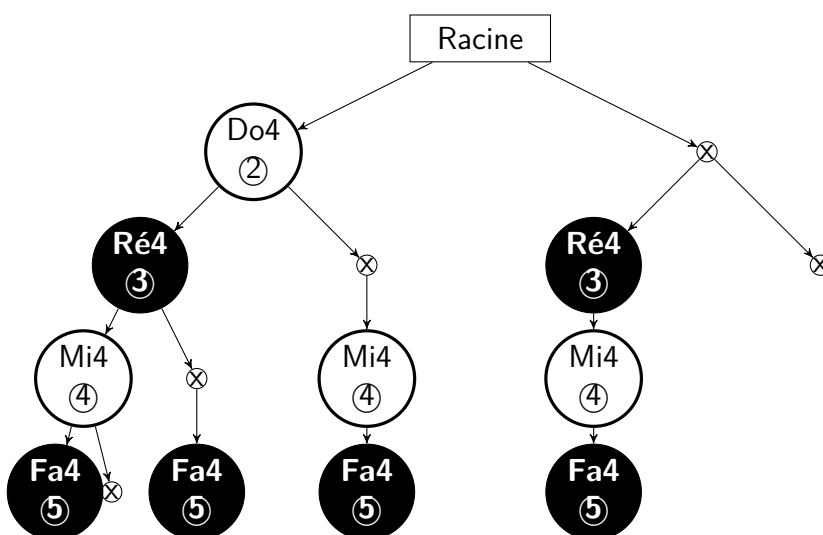


Figure 8.35 Représentation en arbre minimisant le nombre de notes manquantes

Une fois la construction de l'arbre terminée, l'alignement fin peut finalement être trouvé. Cet alignement correspond simplement à la meilleure des branches du dernier niveau, c'est-à-dire la branche ayant le moins de notes manquantes \otimes et, en cas d'ambiguïté, la branche ayant la plus haute moyenne des valeurs de confiances. Les prochaines sous-sections présentent quelques particularités du bloc d'alignement fin.

8.3.1 Notes jouées en polyphonie

L'exemple utilisé précédemment pour présenter le fonctionnement du bloc d'alignement fin était basé sur une partition de musique simpliste composée de quatre notes jouées en monophonie, c'est-à-dire jouées l'une à la suite de l'autre. En présence de notes jouées en polyphonie telles que pour l'extrait présenté à la figure 8.3 et résumé dans le tableau 8.7, l'arbre serait simplement construit tel qu'illustré à la figure 8.36. La principale différence est que les notes jouées en polyphonies sont regroupées par un cercle en pointillé pour indiquer que l'ordre d'apparition dans le temps peut être inversé. Par exemple, pour le cas de la figure 8.36, le début de la note Sol#4 peut-être situé dans le temps avant ou après le début de la note Si4.

Tableau 8.7 Exemple de notes représentant une partition de musique avec des notes jouées en polyphonie

Notes	Si3/Sol#4	La4	Si4	Do5
-------	-----------	-----	-----	-----

8.3.2 Notes potentielles de même départ et de même identité

En cas de doute sur la position de la fin d'une note, la couche de transcription automatique présentée au chapitre 7 peut enregistrer plusieurs notes potentielles de même identité et de même moment de début, mais de moment de fin différent. En présence de ces notes, seule la note ayant la meilleure compatibilité musicale au niveau de son moment de fin est retenue pour la construction de l'arbre. Ce choix permet simplement de limiter davantage la croissance de l'arbre.

8.3.3 Correction d'imperfections

L'alignement fin extrait de l'arbre peut parfois contenir quelques imperfections pouvant être corrigées a posteriori. Ces imperfections dispersées peuvent résulter du fait que certaines branches de l'arbre ne soient pas entièrement construites. Ces imperfections peuvent aussi résulter du fait que l'arbre soit construit sous l'hypothèse que les notes jouées sont dans le bon ordre séquentiel. Bien que ce soit généralement le cas, il peut arriver que la couche de transcription automatique détecte une note de musique plus tôt que le moment où elle a réellement commencé et, ainsi, produire le déclassement de l'ordre joué.

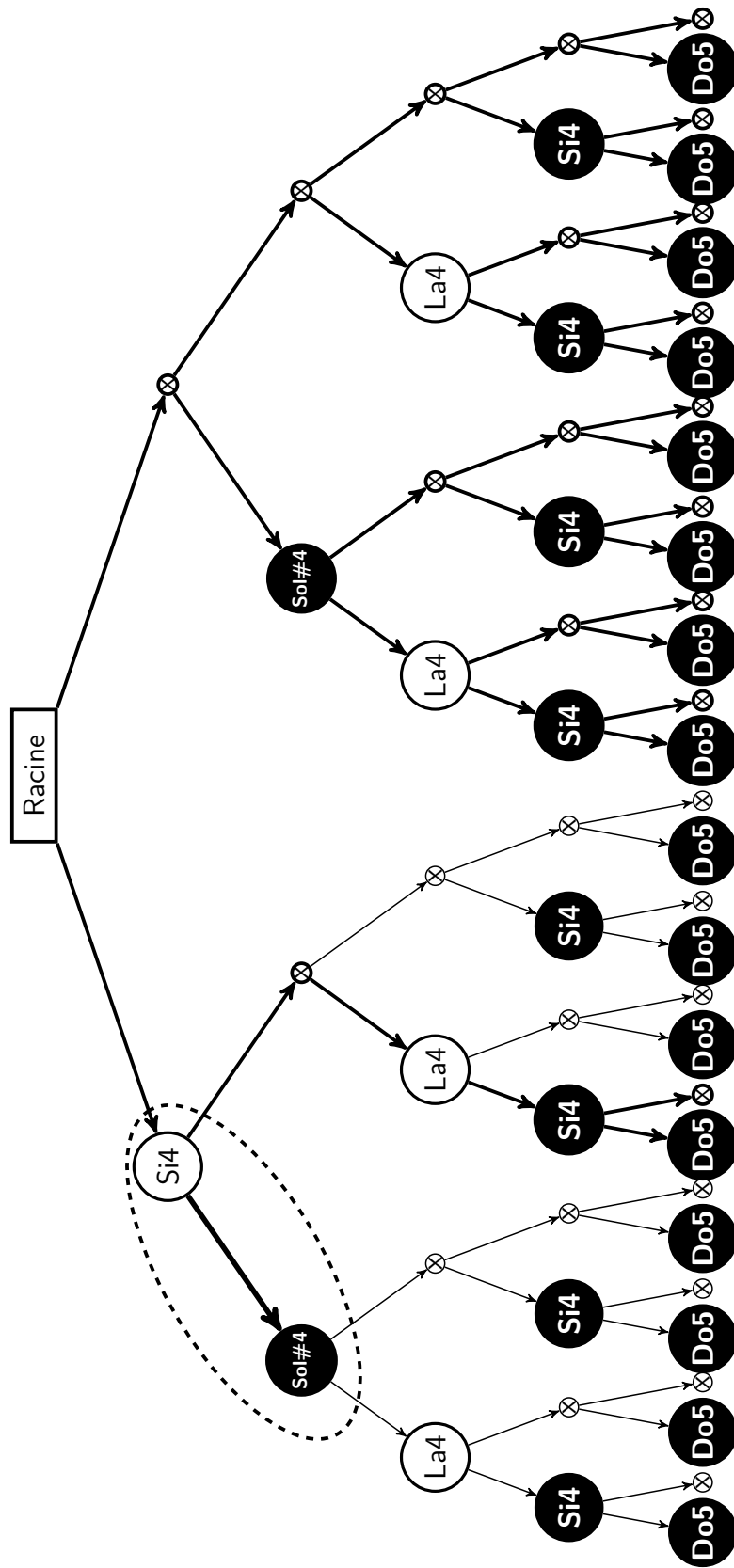


Figure 8.36 Représentation en arbre d'un exemple avec polyphonie

Afin de corriger ces imperfections, lorsqu'une ou plusieurs notes consécutives de la partition de musique ne sont pas alignées, mais que les notes autour d'elles le sont, l'algorithme repère pour chacune des notes non alignées, toutes les notes estimées par la couche de transcription automatique, dont leur identité (ex. Ré4) correspond à l'identité de la note de la partition non alignée en question et où leur corps chevauche la plage en secondes délimitée par le début de la note alignée à gauche et le début de la note alignée à droite tel qu'illustré dans l'exemple de la figure 8.37. Une fois ces notes repérées, la note dont le début est le plus près du début attendu pour cette note est alignée à la note de la partition en question. Il est à noter que pour les résultats du chapitre 12, ces corrections sont effectuées que pour les cas où le nombre de notes consécutives et non alignées est inférieur à quatre.

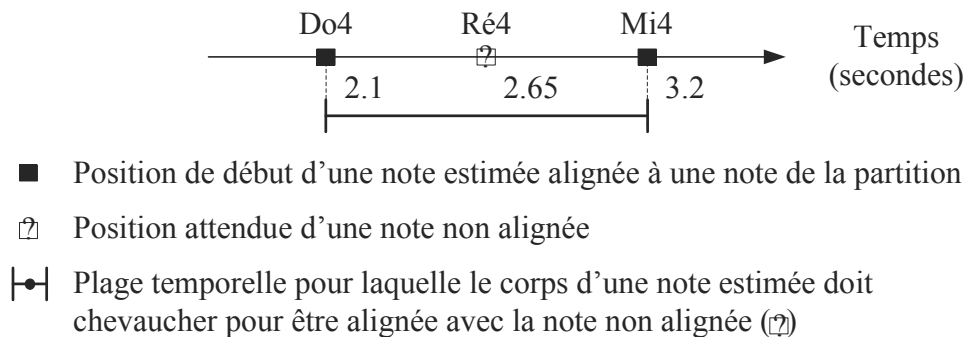


Figure 8.37 Plage temporelle pour la correction d'imperfections

8.4 Contributions scientifiques de la couche

La couche d'alignement est probablement la couche la plus riche en contributions scientifiques. Puisqu'il n'existe actuellement aucun algorithme polyphonique d'alignement de la partition de musique capable de donner autant de liberté au musicien. En effet, pour fonctionner correctement, les algorithmes existants doivent préalablement et minimalement connaître le numéro de mesure de la partition de musique où le musicien commencera à jouer. Pour sa part, la couche d'alignement proposée peut retrouver le musicien dans la partition, et ce, sans même savoir à priori ce que le musicien jouera. Pour ce faire, il suffit que le musicien joue un segment suffisamment long pour que le bloc d'alignement brut puisse retrouver le segment en question. La couche d'alignement a aussi été conçue pour assurer une flexibilité relativement au tempo, puisqu'elle peut fonctionner même si le musicien joue à un tempo légèrement différent et même si, durant certains passages, le musicien accélère ou décélère. De plus, comme il sera vu au chapitre 12, la couche d'alignement a aussi été conçue pour être très robuste en présence de performances musi-

cales jouées par des musiciens amateurs, c'est-à-dire des performances contenant plusieurs erreurs de performances musicales.

Contrairement aux algorithmes d'alignement conventionnels qui utilisent généralement pour faire l'alignement des éléments de traitement du signal tels que la tonie, la couche d'alignement utilise des notes composées d'un moment de début, d'un moment de fin, d'une tonie moyenne et d'une intensité perceptuelle. Le fait d'utiliser des notes permet de créer un niveau d'abstraction supérieur entre la couche d'alignement et la couche de traitement du signal. D'ailleurs, cette façon de faire ressemble davantage à ce qu'un musicien ferait pour suivre un autre musicien à travers une partition de musique, puisque ce dernier entend des notes et non pas des données de traitement du signal. Il est à noter aussi que le fait d'utiliser des notes composées de plusieurs tonies au lieu d'utiliser directement les tonies de chacune des trames d'échantillons audio permet à la couche d'alignement de faire moins de comparaisons pour trouver l'alignement final puisqu'il y a moins de notes que de trames dans un signal audio à traiter et, par conséquent, plus de ressources CPU disponibles pour effectuer les comparaisons. D'ailleurs, cette particularité permet d'obtenir suffisamment de ressources CPU pour donner la liberté au musicien de commencer là où il le désire dans la partition de musique, et ce, sans même avoir à préalablement informer le système de ses intentions.

Pour terminer, comparativement à l'algorithme monophonique d'alignement développé lors de la maîtrise [22] qui détient des caractéristiques équivalentes quant à la liberté sur le jeu, la robustesse en présence d'erreurs de performance et la rentabilisation des ressources CPU et mémoire, l'algorithme d'alignement développé lors de ce projet de doctorat a été conçu pour fonctionner en présence de signaux et de partitions de musique polyphoniques. Opportunément, cette évolution a eu pour effet d'améliorer la performance en présence de partitions monophoniques. En effet, bien qu'une partition monophonique soit composée que de notes jouées séquentiellement, le signal audio correspondant à la performance du musicien est quant à lui sujet à la polyphonie puisque, pour de multiples raisons telles que la réverbération et résonance des cordes adjacentes, les notes jouées précédemment par le musicien peuvent significativement chevaucher dans le temps les nouvelles notes jouées, et ce, tel qu'il sera discuté à la section 12.1 du chapitre 12.

Le prochain chapitre décrit le fonctionnement de la couche comparative qui a pour objectif d'estimer les erreurs de performances du musicien telles que les notes jouées trop tôt ou trop tard et les erreurs de justesse.

CHAPITRE 9

COUCHE COMPARATIVE

Ce chapitre présente le fonctionnement de la couche comparative utilisée par le module *Analyse de la performance musicale* de la figure 5.2. La couche comparative utilise comme entrées les variables de configuration et l’alignement de sortie de la couche d’alignement. Cette couche permet de générer la sortie du module *Analyse de la performance musicale*, c’est-à-dire une liste de résultats de comparaison entre la performance musicale du musicien et la partition de musique.

La couche comparative fonctionne sous l’hypothèse que chacune des notes estimées ayant été associée à une note de la partition de musique, et ce, par le biais de la couche d’alignement, correspond à une note jouée par le musicien. Ainsi, en comparant chacune de ces notes estimées à la note de la partition en question, il est possible d’obtenir les trois déviations énumérées ci-dessous :

1. Déviation sur le moment d’attaque des notes jouées ;
2. Déviation sur la durée des notes jouées ;
3. Justesse de l’intonation des notes jouées.

9.1 Difficultés liées aux imperfections des couches inférieures

Présentement, due aux lacunes des diverses couches inférieures, il n’est pas possible de réaliser un système parfait d’alignement de la partition de musique. Par exemple, la couche de traitement du signal peut parfois faillir à détecter certaines tonies nécessaires à la détection de certaines notes jouées par le musicien et, pour sa part, la couche de transcription automatique peut parfois estimer incorrectement la position de début ou de fin de certaines notes produisant ainsi des imperfections au niveau de l’alignement.

Pour ces raisons, il est actuellement impossible pour la couche comparative d’affirmer avec certitude si la présence d’une déviation par rapport à la partition de musique est le résultat de l’imperfection de l’une des couches logicielles inférieures ou simplement le résultat d’une erreur de performance de la part du musicien. Afin de pallier ce problème,

la couche est conçue pour valider chacune de ces déviations à l'aide de techniques variées qui seront présentées ultérieurement dans les sections 9.5 à 9.6 inclusivement.

9.2 Définition des variables

Avant d'entrer dans les détails, il est pertinent de définir, à partir du tableau 9.1, quelques variables et fonctions utilisées tout au long de ce chapitre. Ces variables sont aussi présentées graphiquement à l'aide de l'exemple de la figure 9.1 où, à partir de l'index commun n , il est possible de constater les paires de notes créées par la couche d'alignement.

Tableau 9.1 Définition des variables utilisées par la couche comparative

Variable ou fonction	Définition
n	Index d'une paire de notes formée par la couche d'alignement, c'est-à-dire une note estimée en paire avec une note de la partition jouée
d_n	Position de début en mesures de la note de la partition de musique associée à une paire n
\tilde{d}_n	Position de début en secondes d'une note correctement estimée par la couche de transcription automatique associée à une paire n
f_n	Position de fin en mesures de la note de la partition de musique associée à une paire n
\tilde{f}_n	Position de fin en secondes d'une note correctement estimée par la couche de transcription automatique associée à une paire n
m	Indice d'un groupement d'une ou de plusieurs paires de notes dont les notes de la partition de musique démarrent toutes au même instant
D_m	Position de début en mesures d'un groupement d'indice m (voir la définition de l'indice m ci-dessus)
\tilde{D}_m	Moyenne des positions de début en secondes comprises dans un groupement d'indice m (voir la définition de l'indice m ci-dessus)
$v(n)$	Fonction permettant de trouver l'indice m du groupement D_m ou \tilde{D}_m contenant la paire d'indice n .
$\tau(m)$	Fonction permettant de trouver le tempo instantané en mesures par seconde à la position m correspondant à un groupement de paires de notes

9.3 Estimation du tempo

Le tempo est une mesure nécessaire pour évaluer certains résultats de sortie tels que la déviation sur le moment de l'attaque d'une note de musique. Il est à noter que le tempo peut varier d'un endroit à l'autre dans la partition de musique puisque le musicien peut à tout moment accélérer ou décélérer, et ce, volontairement ou involontairement. Ainsi, le tempo instantané $\tau(m)$ en mesures par seconde est calculé à l'aide de l'équation 9.1, et ce, à la position d'un groupement de notes d'indice m .

$$\tau(m) = \frac{D_m - D_{m-1}}{\tilde{D}_m - \tilde{D}_{m-1}} \quad (9.1)$$

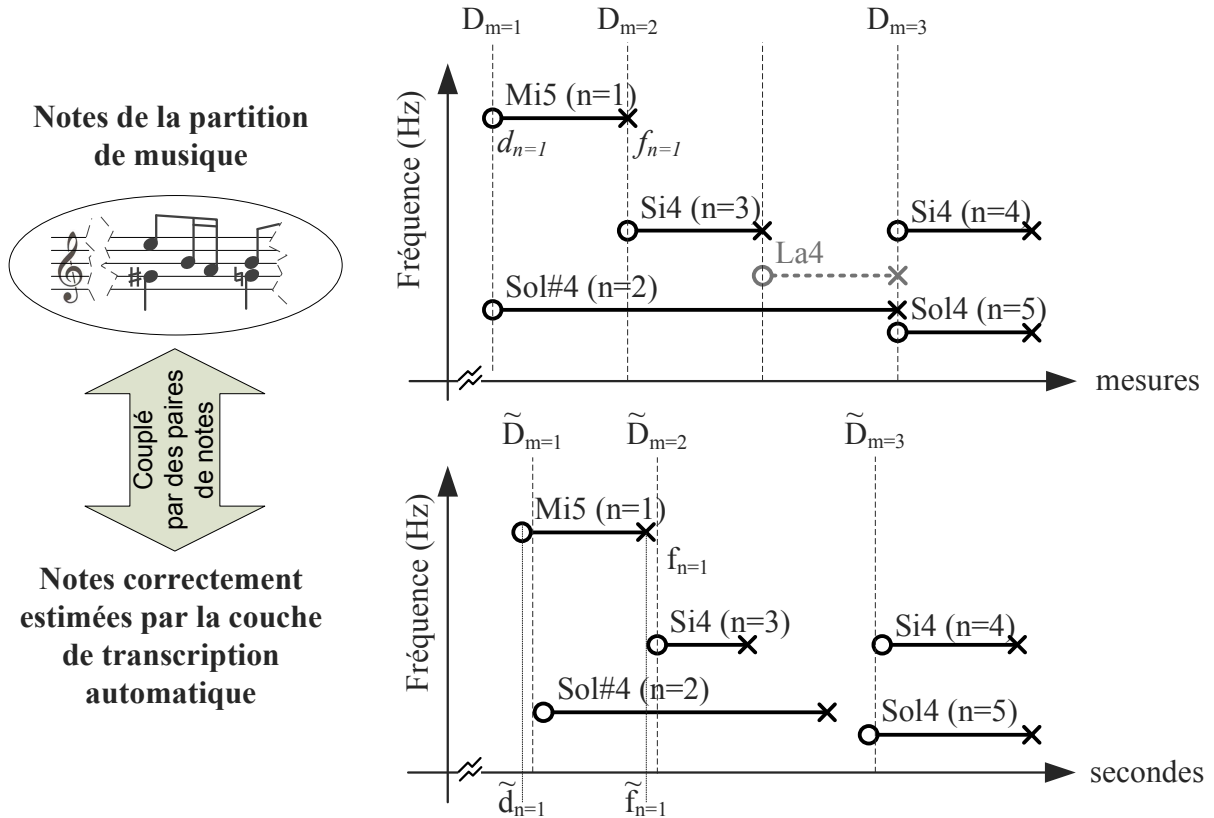


Figure 9.1 Variables utilisées par la couche comparative

9.4 Déviation sur le moment de l'attaque

En référence avec la partition de musique, un musicien peut jouer une note trop tôt ou trop tard. Cette déviation par rapport à la partition de musique fait partie des rétroactions attendues d'un professeur virtuel de musique et, ainsi, fait partie des diverses sorties de la couche. Cette déviation a_n en secondes, pour une paire de notes d'indice n , est calculée à l'aide de l'équation 9.2.

$$a_n = \min_{i=1,2,\dots,I} \left[\underbrace{(\tilde{d}_n)}_{\text{Position réelle}} - \underbrace{\left(\tilde{D}_{v(n)-i} + \frac{D_{v(n)} - D_{v(n)-i}}{\tau(v(n) - i)} \right)}_{\text{Position attendue}} \right] \quad (9.2)$$

En bref, pour la note dont il est question, cette équation évalue la différence minimale entre la position de début réelle et diverses valeurs estimées de la position de début attendue. Pour obtenir chacune des positions attendues, l'équation base simplement ses calculs sur les I notes précédentes de l'alignement où I a été fixé à quatre pour les résultats présentés au chapitre 13. À l'aide de l'opérateur minimum, la valeur de déviation de sortie correspond

au cas où la position estimée est la plus près de la valeur réelle. L'objectif derrière cette technique est de s'assurer que la déviation obtenue ne soit pas basée sur des notes dont le début fut mal estimé par la couche de transcription automatique. Bien que la déviation minimale peut parfois ne pas être la déviation optimale, ce choix se justifie par la stratégie suivante :

Afin d'éviter de distraire inutilement le musicien, il est jugé préférable de ne pas informer ce dernier d'une erreur de performance qu'il a possiblement faite, et ce, lorsque le système a des doutes raisonnables de croire qu'il puisse s'agir, en fait, d'une erreur due aux imperfections des diverses couches logicielles.

Il est à noter que malgré qu'il puisse s'agir effectivement d'une erreur de la part du musicien et non d'une imperfection du système, il est raisonnable d'affirmer que si cette erreur est répétée de façon récurrente par le musicien, elle sera éventuellement transmise par la couche comparative et, par conséquent, sera éventuellement révélée au musicien.







9.5 Déviation sur la durée

En référence avec la durée d'une note prescrite par la partition de musique, un musicien peut maintenir une note trop longtemps ou pas assez longtemps. Cette déviation fait aussi partie des diverses sorties de la couche. Cette déviation r_n en secondes, pour une paire de notes d'indice n , est calculée à l'aide de l'équation 9.3. Comme la durée d'une note de la partition dépend aussi du type d'accent avec lequel elle doit être jouée, le facteur Ω a été ajouté à l'équation. La valeur de Ω pour un accent donné peut être trouvée à partir du tableau 9.2.

$$r_n = \min_{i=1,2,\dots,I} \left[\underbrace{(\tilde{f}_n - \tilde{d}_n)}_{\text{Durée réelle}} - \underbrace{\left(\frac{\Omega (f_n - d_n)}{\tau(v(n) - i)} \right)}_{\text{Durée attendue}} \right] \quad (9.3)$$

En bref, cette équation calcule la différence minimale entre la durée réelle et diverses valeurs estimées de la durée attendue. Comme on peut le remarquer, cette équation est en plusieurs points similaire à l'équation 9.2. Ceci est dû au fait qu'elle partage la même stratégie visant à ne pas distraire inutilement le musicien en cas de doute sur la validité d'une déviation.

Tableau 9.2 Durée d'une note jouée en fonction de l'accent

A_i	Symbole	Valeur de Ω	Type d'accent	Durée nominale de la note
1		1	Portamento	Inchangé
2		1	Détaché	Inchangé
3		0.5	Staccato	Moitié de la durée nominale
4		0.25	Staccatissimo	Quart de la durée nominale
5		1	Marcato	Inchangée
6		1	Martelato	Inchangée

9.6 Justesse de l'intonation

Par rapport à la tonie nominale, un musicien peut jouer une note trop aiguë ou trop grave. Cette déviation est identifiée par le terme *justesse de l'intonation* et fait partie des diverses sorties de la couche comparative. Cette déviation j_n en cents, pour une paire de notes d'indice n , est calculée à l'aide de l'équation 9.4 où a_n correspond à la tonie de la note jouée et b_n à la tonie nominale de la note de la partition de musique. Il est à noter que l'unité de mesure utilisée pour cette déviation est le cent puisque, comparativement au hertz, le cent permet d'obtenir une échelle linéaire pour mesurer la distance entre des notes, c'est-à-dire une échelle dont les notes adjacentes sont séparées entre elles par une distance constante de 100 cents.

$$j_n = 1200 \log_2 \left(\frac{a_n}{b_n} \right) \quad (9.4)$$

La tonie d'une note jouée varie généralement dans le temps tel que c'est le cas en présence d'un vibrato. Il a d'ailleurs été observé que, pour le cas d'un vibrato, la tonie globalement perçue se rapproche de la moyenne géométrique des tonies individuelles contenues dans la note en question et que les résultats obtenus avec la moyenne arithmétique sont quasi identiques à ceux obtenus à partir de la moyenne géométrique [74]. Par contre, pour des notes de courte durée, où le vibrato contient moins de deux périodes, la tonie globalement perçue dépend davantage des tonies individuelles obtenues vers la fin du vibrato [10].

Par conséquent, la tonie d'une note jouée est trouvée à l'aide de deux modèles différents, c'est-à-dire un modèle pour les notes de courte durée selon l'équation 9.5 et un modèle pour les autres notes selon l'équation 9.6. Il est à noter que le premier modèle est une version discrétisée du modèle proposé par l'article [10], ce modèle a pour objectif de pondérer

chacune des tonies individuelles, et ce, dans le but de favoriser celles situées vers la fin de la note.

$$a_n = \frac{\sum_{i=0}^{I-1} (e^{-\alpha(I-i)\Delta t} + \beta) p_n[i]}{\sum_{i=0}^{I-1} (e^{-\alpha(I-i)\Delta t} + \beta)} \quad (9.5)$$

$$a_n = \frac{1}{I} \sum_{i=0}^{I-1} p_n[i] \quad (9.6)$$

Pour ces deux modèles, la variable $p_n[i]$ correspond à la i^e tonie parmi les I tonies instantanées de la note jouée appartenant à la paire de notes d'indice n pour laquelle les tonies sont séparées entre elles par une distance constante de Δt secondes. Pour leur part, le paramètre β correspond à une variable pouvant être interprétée comme « la quantité de moyennes temporelles à long terme » [10] et le paramètre α correspond à une variable pouvant être interprétée comme « la vitesse de décroissance de la fonction exponentielle » [10]. À titre d'exemple, la figure 9.2 démontre la forme de la courbe de pondération $(e^{-\alpha(I\Delta t - i\Delta t)} + \beta)$ pour diverses valeurs de α et pour une valeur de β fixée à 0.2. Il est à noter que pour les résultats du chapitre 13, les paramètres α et β sont fixés à 22 et 0.2 respectivement, et ce, tels que recommandés par les créateurs du modèle de l'équation 9.5 et une note jouée est jugée comme étant de courte durée si sa durée est inférieure à 130 millisecondes.

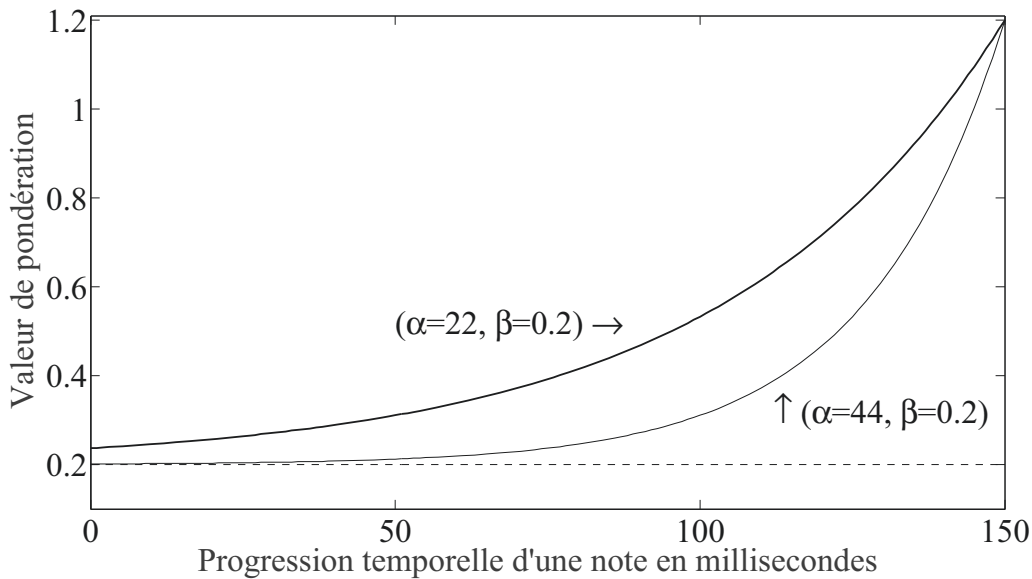


Figure 9.2 Courbe de pondération pour une note de 150 ms

9.7 Notes manquantes

Les notes manquantes correspondent aux notes de la partition de musique qui n'ont pas été jouées par le musicien. Leur détection relève des tâches d'un professeur virtuel de musique puisque le fait de sauter une note de la partition de musique constitue une erreur de performance de la part du musicien.

Afin d'estimer les notes manquantes, cette couche fonctionne sous l'hypothèse que toutes les notes jouées par le musicien ont été détectées par la couche de transcription automatique et que l'alignement a lié correctement chacune de ces notes aux notes de la partition. Rappelons que la couche de traitement du signal, la couche de transcription automatique et la couche d'alignement ont toutes été conçues dans cette optique. Par conséquent, une note manquante correspond simplement à une note de la partition de musique n'ayant pas été alignée par la couche d'alignement.

Les prochains chapitres décrivent la performance de chacune des quatre couches du système proposé dont le chapitre 13 correspond à la performance de cette dernière couche.

CHAPITRE 10

PERFORMANCE : COUCHE DE TRAITEMENT DU SIGNAL

Ce chapitre présente les résultats de performance obtenus pour la couche de traitement du signal présentée sommairement au chapitre 5 et en détail au chapitre 6, soit la couche responsable d'estimer l'ensemble des tonies contenues dans chacune des trames du signal audio numérique d'entrée du système proposé. Il est à noter que les paramètres de configuration, les critères d'évaluation, la démarche ainsi que les signaux audio utilisés pour obtenir les résultats sont présentés aux sections 6.2 et 6.4 du chapitre 6.

Tel que discuté à la section 6.5.5 du chapitre 6, la performance de l'estimateur de tonies proposé dépend du seuil appliqué sur chacun des niveau de confiance. Par conséquent, les résultats sont représentés graphiquement selon les figures 10.1 et 10.2 pour les signaux de violon monophoniques et polyphoniques respectivement. Il est noter que le segment de signal audio correspondant à la variation 5 de la sonate a été exclue pour la figures 10.1, comme ce segment audio contient des notes jouées en polyphonie.

Les graphiques des figures 10.1 et 10.2 contiennent deux courbes associées à deux axes d'ordonnées différents où l'axe des ordonnées à gauche correspond au pourcentage de tonies estimées avec succès et l'axe des ordonnées à droite correspond au pourcentage de tonies estimées en trop. Ainsi, à titre d'exemple, un pourcentage de tonies estimées en trop de 200% impliquerait qu'il y a eu deux fois plus de tonies estimées en erreur que de tonies nominales à estimer. Quant à elle, l'abscisse correspond à un seuil de conservation appliqué sur le niveau de confiance, c'est-à-dire un seuil où toutes les tonies ayant un niveau de confiance inférieur à cette valeur sont supprimées. Les deux prochaines sections présentent en détail les résultats obtenues pour des signaux monophoniques et polyphoniques.

10.1 Performance pour des signaux monophoniques

Le tableau 10.1 compare, pour des signaux monophoniques, la performance de l'estimateur de tonies proposé par rapport à d'autres estimateurs de tonies dont deux ont été conçus spécifiquement pour le violon, soit l'estimateur de tonies utilisé par le système d'alignement nommé *Fuzzy Alignment of Musical Score (FAMS)* [22] en anglais et l'estimateur de tonies

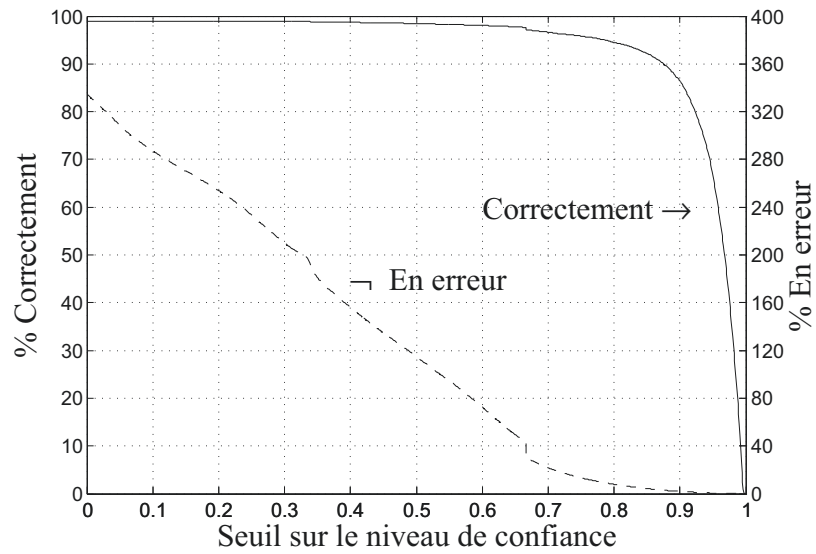


Figure 10.1 Pour le cas d'un signal monophonique, pourcentage de tonies estimées correctement et en erreur selon le seuil utilisé sur le niveau de confiance

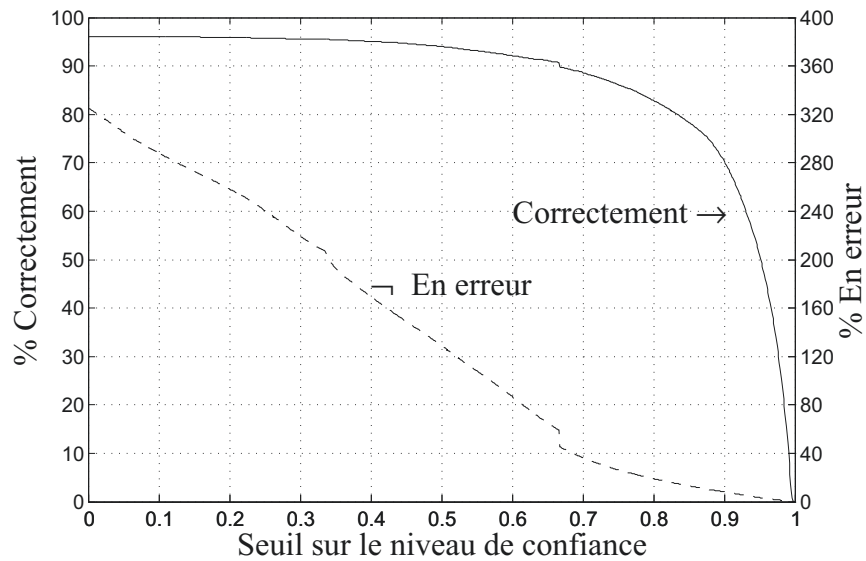


Figure 10.2 Pour le cas d'un signal polyphonique, pourcentage de tonies estimées correctement et en erreur selon le seuil utilisé sur le niveau de confiance

de la référence [31] nommé *Violin Transcriber for Personalized Learning* (VMTPL) en anglais.

Tableau 10.1 Performance de l'estimateur de tonies pour des signaux monophoniques

No.	Estimateur de tonies	% Correctement	% En erreur
1	YIN [9]	86.5	11.1
2	VMTPL [31] (dédié au violon)	88.1	20.2
3	FAMS [22] (dédié au violon)	89.2	11.7
4	Méthode proposée (Seuil=0.88)	89.3	2.6
5	Méthode proposée (Seuil=0.71)	96.5	20.2

Selon le tableau 10.1, l'estimateur de tonies proposé surpasse les trois autres estimateurs de tonies. Par exemple, en appliquant un seuil sur le niveau de confiance global qui permet d'obtenir le même pourcentage de tonies estimées correctement que le meilleur des estimateurs de comparaison, soit FAMS [22] à environ 89,2 %, l'estimateur de tonies proposé donne un pourcentage inférieur de tonies estimées en erreur, c'est-à-dire 11,7 % pour le FAMS contre 2,6 % pour l'estimateur proposé. De plus, par rapport à l'unique estimateur polyphonique spécialisé pour le violon [31], pour un même pourcentage de tonies estimées en erreur, soit 20,2 %, l'estimateur de tonies proposé donne un pourcentage plus élevé de tonies estimées correctement, c'est-à-dire 88,1 % pour le VMTPL contre les 96,5 % pour l'estimateur proposé.

10.2 Performance pour des signaux polyphoniques

Le tableau 10.2 présente maintenant la performance de l'estimateur de tonies proposé en présence de signaux audios polyphoniques. Comme l'estimateur polyphonique de référence [31] ne fournit aucune solution pour déterminer si plus d'une tonie est présente dans une trame de signal audio, le nombre total de tonies estimées par trame a été fixé à un et deux pour les résultats no. 3 et no. 4 respectivement.

À partir du tableau 10.2, il est possible de remarquer que les estimateurs monophoniques de tonies YIN et FAMS ne conviennent plus pour des signaux audio polyphoniques de violon. Ceci s'explique par le fait qu'ils sont basés sur des paramètres temporels où les tonies sont estimées en trouvant la périodicité dans le signal temporel. En fait, ces estimateurs de tonies sont généralement bons pour des signaux monophoniques étant donné que la période

Tableau 10.2 Performance de l'estimateur de tonies pour des signaux polyphoniques

No.	Estimateur de tonies	% Correctement	% En erreur
1	YIN [9] (monophonique)	69.8	21.4
2	FAMS [22] (monophonique)	72.9	24.2
3	VMTPL[31] (1 tonie/trame)	75.6	29.7
4	VMTPL[31] (2 tonies/trame)	89.7	120.8
5	Méthode proposée (Seuil=0.76)	85.4	24.1
6	Méthode proposée (Seuil=0.67)	89.7	44.4

de la tonie est généralement bien apparente dans le domaine du temps. Cependant, dans ce domaine, la périodicité de la tonie est beaucoup moins apparente en présence de signaux audio polyphoniques et, par conséquent, ces estimateurs de tonies ne sont habituellement pas très performants pour ce type de signal.

En contre partie, il est possible de remarquer que pour des signaux polyphoniques de violon, l'estimateur de tonies proposé surpasse les estimateurs de tonies monophoniques (YIN et FAMS). En effet, en considérant l'estimateur FAMS pour un pourcentage de tonies en erreur équivalent ($\sim 24.2\%$), l'estimateur de tonies proposé donne un pourcentage plus élevé de tonies estimées correctement, soit 72.9% pour l'estimateur FAMS contre 85.4% pour l'estimateur proposé.

Enfin, on remarque que pour des signaux polyphoniques de violon, l'estimateur de tonies proposé est plus approprié que l'estimateur de tonies polyphoniques dédié pour le violon utilisé comme référence [31] puisque pour le même pourcentage de tonies estimées correctement ($89,7\%$), l'estimateur proposé donne un pourcentage inférieur de tonies estimées en extra, c'est-à-dire $120,8\%$ pour le VMTPL comparativement à $44,4\%$ pour l'estimateur proposé.

10.3 Temps de traitement

En utilisant l'unité centrale de traitement Intel Core 2 Quad Q6600 d'un ordinateur portable et le langage de programmation C++, la couche de traitement du signal proposée prend en moyenne 14 secondes pour traiter une minute de signal audio. Ces résultats dé-

montrent que la complexité de cette couche est suffisamment faible pour être utilisée pour des applications en temps réel.

10.4 Discussion

Selon les résultats obtenus, pour le cas des signaux monophoniques et polyphoniques de violon contenant des notes lentes et très rapides jouées avec divers styles, l'estimateur proposé surpasse tous les estimateurs utilisés aux fins de comparaison. De plus, la complexité de l'estimateur de tonies proposée est également suffisamment faible pour permettre la réalisation d'applications fonctionnant en temps réel. Enfin, en consultant le niveau de confiance lié à chacune des tonies estimées, un système client a la liberté de décider s'il désire utiliser ou non chacune des tonies potentielles estimées.

CHAPITRE 11

PERFORMANCE : COUCHE DE TRANSCRIPTION AUTOMATIQUE

Ce chapitre présente les résultats de performance obtenus pour la couche de transcription automatique présentée sommairement au chapitre 5 et en détail au chapitre 7, soit la couche responsable d'estimer l'ensemble des notes jouées par le musicien, et ce, à partir des tonies estimées par la couche de traitement du signal.

Les signaux audio utilisés pour évaluer la procédure proposée sont les mêmes que ceux utilisés pour la couche de traitement du signal, c'est-à-dire le deuxième mouvement de la deuxième Sonate pour violon de Prokofiev jouée par Gil Shaham [59], et ce, pour les mêmes raisons que celles élaborées à la section 6.2 du chapitre 6.

Bien que la couche ait été conçue pour fonctionner avec la plupart des instruments de musique, le violon a tout de même été sélectionné comme le musicien a plein contrôle sur la hauteur de la tonie et sur l'intensité des notes. En effet, comme la tonie d'une note à travers le temps peut croiser toute la plage attendue et que l'intensité sonore peut prendre différentes formes, le choix de cet instrument de musique permet d'évaluer la couche de transcription automatique en profondeur.

11.1 Apport de la couche par rapport à l'approche typique

Le tableau 11.1 présente les résultats obtenus en terme de pourcentage de la note estimée avec succès. Pour être estimée avec succès, une note doit commencer et se terminer selon la figure 11.1 où Δ est fixé à 50ms. De plus, la note doit être continue, c'est-à-dire qu'une tonie doit être liée à chacune des trames d'échantillons audio. Pour cette évaluation, l'approche typique consiste à lier ensemble toutes les tonies consécutives ayant le même index MIDI en utilisant l'équation 7.1, et ce, tel que discuté à la section 7.1 du chapitre 7.

Il est à noter que ces résultats dépendent de la performance de la couche de traitement du signal. En effet, l'amélioration de la couche de traitement du signal aurait nécessairement un impact positif sur la performance de la couche de transcription automatique.

Tableau 11.1 Performance de l'approche typique versus proposée, (a) sans l'étape *fusionner les adjacents*, (b) avec l'étape *fusionner les adjacents*

Approche	Paramètres	% Notes estimées correctement
Typique	$p_{ref} = 440.0Hz$ $I_{ref} = 69$	75.5%
Typique	$p_{ref} = 444.7Hz$ $I_{ref} = 69$	81.0%
Proposée	sans <i>fusionner adj.</i>	(a) 83.5%
Proposée	avec <i>fusionner adj.</i>	(b) 89.4%

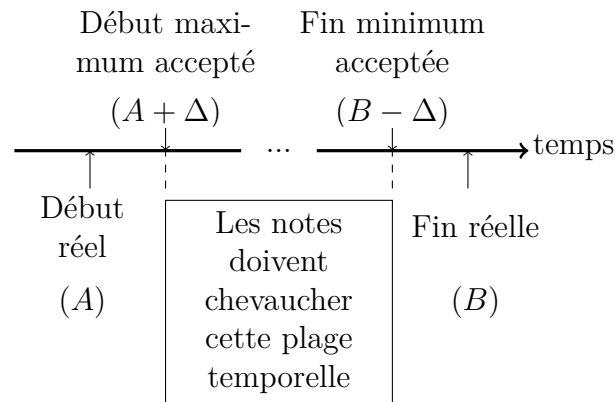


Figure 11.1 Condition pour une note estimée correctement

Selon ces résultats, la procédure proposée surpasse l'approche typique, car même lorsque la tonie de référence (p_{ref}) est connue avec précision, la performance de la procédure proposée demeure supérieure.

Afin de démontrer l'importance de l'étape *Fusionner les adjacents*, les résultats sont présentés avec et sans cette étape. En raison du pourcentage significatif de notes estimées avec succès (83,5% à 89,4%), ces résultats démontrent clairement que l'étape contribue bel et bien à la récupération de nombreuses tonies manquantes.

11.2 Décomposition en plusieurs notes

Comme discuté à la sous-section 7.4 du chapitre 7, en raison de la réverbération, de l'écho et de l'anticipation du musicien, la position réelle du début et de la fin d'une note peuvent être légèrement différentes de celles estimées. En se référant à la sous-section 7.4.1 du chapitre 7, la stratégie proposée pour résoudre ce problème consiste à enregistrer chaque possibilité de note dans la *Liste de notes*. Le tableau 11.2 présente les résultats de performance avec et sans cette stratégie en utilisant les critères de performance de la figure 11.2 au lieu de la figure 11.1 où les critères sont maintenant plus restrictifs pour la position de début et

de fin d'une note estimée avec Δ_1 , Δ_2 , Δ_3 et Δ_4 correspondant respectivement à 200ms, 50ms, 50ms et 350ms. Il est à noter que pour être considérée comme une note, la durée de la note doit être supérieure ou égale à 58ms.

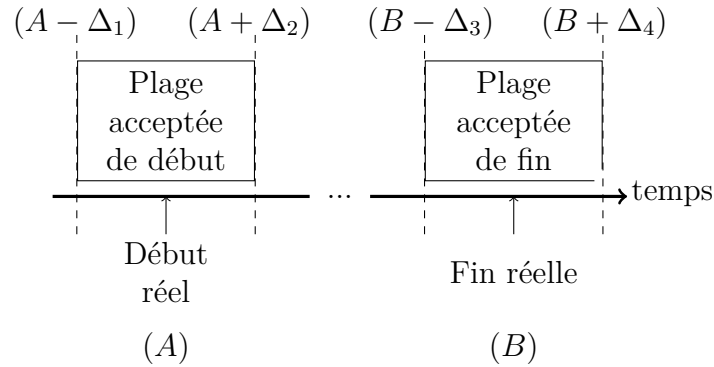


Figure 11.2 Condition plus restreinte pour une note estimée correctement

Tableau 11.2 Impact de la décomposition en plusieurs notes

Approche	% Estimées correctement	% Estimées en trop
Typique sans la stratégie (444.7Hz)	70.2	445
Proposée sans la stratégie	70.4	324
Proposée avec la stratégie	85.0	487

Il est noter que quatre tonies par trame sont estimées, et ce, même lorsqu'une seule tonie est réellement présente. Cela entraînera, bien entendu, un pourcentage élevé de notes estimées en trop. Notez que ces notes en trop correspondent principalement à des notes en erreur d'octave.

Ces résultats démontrent que de nombreuses notes auront une mauvaise estimation de début et de fin de note si la procédure typique ou la procédure proposée sont utilisées sans la décomposition en plusieurs notes. En contrepartie, la stratégie proposée permet d'augmenter le pourcentage de notes estimées avec succès (+14,6%), mais permet aussi d'augmenter le pourcentage de notes estimées en trop (+163%). La sous-section suivante présente les résultats de performance de l'une des méthodes utilisées pour réduire le pourcentage de notes estimées en trop, soit la suppression de note en fonction de l'intensité sonore.

11.3 Rejet d'une note selon son intensité sonore

Comme discuté dans la sous-section 7.3 du chapitre 7, le pourcentage de notes en trop peut être réduit en rejetant les notes estimées ayant une intensité moyenne sous un seuil

défini. Le tableau 11.3 présente les performances de cette étape de réjection où le seuil a été réglé à 0,13.

Tableau 11.3 Impact du rejet selon l'intensité

Approche	% Estimées correctement	% Estimées en trop
Sans rejet selon l'intensité	85.0	487
Avec rejet selon l'intensité	85.0	285

Ces résultats démontrent que le pourcentage de notes estimées en trop peut être réduit de manière significative avec le rejet selon l'intensité sonore, et ce, sans aucun effet sur le pourcentage de notes estimées avec succès. En outre, à ce stade, la procédure proposée surpasse de nouveau la procédure typique puisque le pourcentage de notes estimées avec succès est 14,8% supérieur et le pourcentage de notes supplémentaires estimées est 160% inférieur.

Il est à noter que si la suppression avait été effectuée au niveau de la tonie au lieu de la note, c'est-à-dire en rejetant chaque tonie dont l'intensité sonore aurait été sous un seuil défini, le pourcentage de notes estimées correctement serait de 82,7% au lieu de 85%, et ce, pour un seuil qui serait ajusté pour obtenir le même pourcentage de notes supplémentaires, soit 285%.

Finalement, même si le pourcentage de notes supplémentaires est considérablement réduit avec le rejet selon l'intensité sonore, il demeure élevé. La sous-section suivante présente les résultats de performance en utilisant les différents niveaux de confiance de note pour réduire davantage le pourcentage de notes estimées en trop.

11.4 Performance des niveaux de confiance de note

La performance des trois niveaux de confiance présentés à la section 7.4 du chapitre 7 est d'abord présentée à l'aide des trois graphiques des figures 11.3, 11.4 et 11.5. Il est à noter que ces graphiques contiennent deux courbes associées à deux axes d'ordonnées différents où l'axe des ordonnées à gauche correspond au pourcentage de notes estimées avec succès et l'axe des ordonnées à droite correspond au pourcentage de notes estimées en trop. L'abscisse correspond à un seuil de conservation appliqué sur le niveau de confiance, c'est-à-dire un seuil où toutes les notes ayant un niveau de confiance inférieur à cette valeur sont supprimées.

De ces trois graphiques, il peut être observé que chacun de ces niveaux de confiance peut être utilisé pour réduire le pourcentage de notes estimées en trop. En fait, chacun de ces

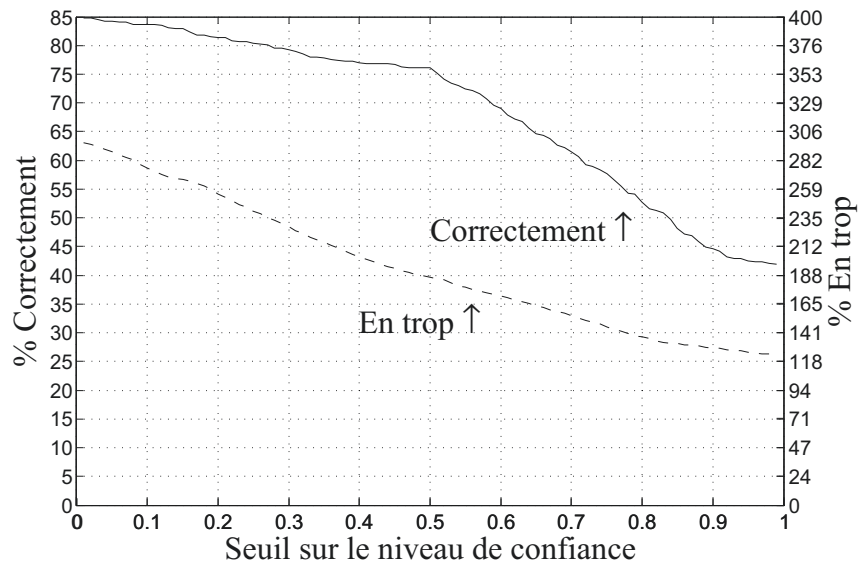


Figure 11.3 Performance du niveau de confiance de début et de fin

niveaux de confiance a été conçu pour atteindre des buts différents. D'abord, le niveau de confiance pour le début et la fin des notes est utilisé pour aider un module client à décider quelle note conserver dans les cas où plusieurs notes auraient le même index midi et se chevaucheraient dans le temps. Pour sa part, le niveau de confiance selon l'octave est utilisé pour aider un module client à décider quoi faire dans le cas où, au même moment, plus d'une note est différente d'une ou de plusieurs octaves. Enfin, la valeur de confiance sur les tonies contenues dans la note peut être utilisée pour sélectionner uniquement les meilleurs candidats pour réaliser des opérations d'approximation telles que c'est le cas pour le calcul de l'alignement brut décrit au chapitre 8.

11.5 Performance des trois niveaux de confiance combinés

Cette sous-section démontre qu'il est possible de réduire davantage le nombre de notes estimées en trop en utilisant un à un chacun des trois niveaux de confiance individuels. Il est à noter que les résultats obtenus pour chacune des étapes de suppression sont présentés dans le tableau 11.4.

Initialement, en utilisant le rejet selon l'intensité sonore, le pourcentage de notes estimées avec succès (EAS) est de 85.0% avec 285% de notes estimées en trop (EET). Ensuite, en rejetant celles qui ont un niveau de confiance c_t sur les tonies inférieur à 0,65, le pourcentage EAS et EET passe maintenant à 81,6% et 63,4% respectivement, c'est-à-dire les résultats

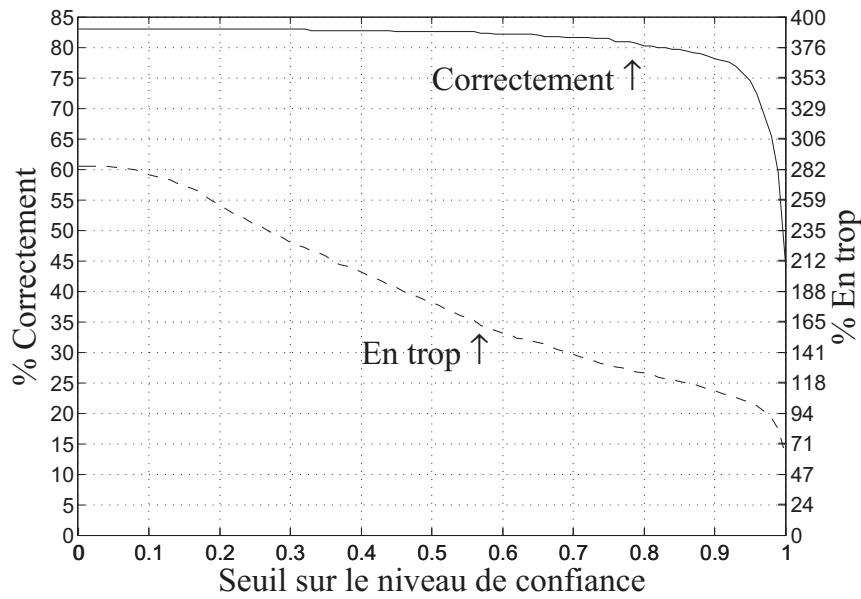


Figure 11.4 Performance du niveau de confiance sur l'octave

Tableau 11.4 Combinaison des trois niveaux de confiance

Étape	Description du moyen de rejet utilisé	% Estimées correctement	% Estimées en trop
Init.	Intensité	85.0	285
1	Confiance des tonies $c_t < 0.65$	81.6	63.4
2	Confiance sur l'octave $(c_n^{max} - c_n) > 0.06$	81.4	58.8
3	Confiance début/fin $(c_n^{max} - c_n) > 0.7$	80.6	52.0

de l'étape 1 du tableau 11.4. Par conséquent, en utilisant simplement le niveau de confiance sur les tonies, il est possible de réduire de manière significative le pourcentage EET, soit de -220,6%.

En supprimant maintenant toutes les notes estimées susceptibles d'être en erreur d'octave, et ce, en supprimant toutes les notes qui se chevauchent dans le temps avec d'autres notes de différents d'octave et qui ont un niveau de confiance c_n d'octave de 0.06 inférieur au meilleur niveau de confiance c_n^{max} d'octave des notes comparées. Cette nouvelle suppression permet d'obtenir 81,4% EAS et 58,8% EET, c'est-à-dire les résultats de l'étape 2 du tableau 11.4.

Finalement, en ne conservant que les notes estimées qui semblent avoir de bonnes positions de début et de fin, et ce, en rejetant toutes les notes qui se chevauchent dans le temps avec

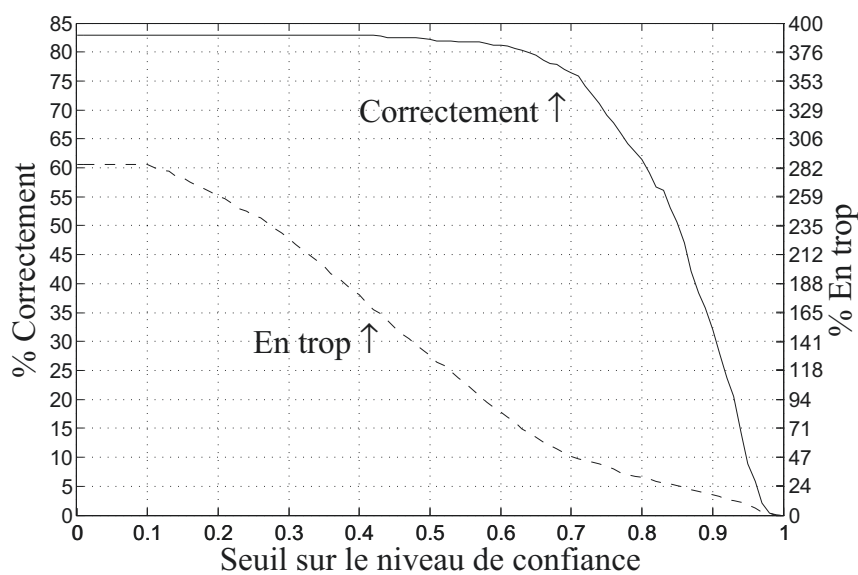


Figure 11.5 Performance du niveau de confiance selon les niveaux de confiance de tonies contenues dans la note

d'autres notes de même index midi qui ont un niveau de confiance c_a de début ou de fin de 0,75 inférieur au meilleur niveau de confiance c_a^{max} de début ou de fin respectivement des notes comparées. Cette nouvelle suppression permet d'obtenir 80,6% EAS et 52% EET, c'est-à-dire les résultats de l'étape 3 du tableau 11.4. Par conséquent, cet exemple démontre qu'en combinant les niveaux de confiance, il est possible de réduire considérablement le pourcentage d'EET, et ce, avec un effet minimal sur le pourcentage de EAS.

11.6 Temps de traitement

En utilisant un ordinateur personnel muni d'une unité centrale de traitement Intel Core 2 Quad Q6600 de 2.4GHz sous le système d'exploitation Windows 7, le langage de programmation C++ et les fils (*threads*, en anglais) pour le parallélisme, la couche de traitement du signal et la couche de transcription automatique nécessitent environ 18 secondes pour traiter une minute de signal audio échantillonné à 44.1KHz, et ce, en utilisant des trames distancées entre elles par 512 échantillons. Il est à noter que l'estimateur de tonies prend environ 10 secondes par minute pour estimer toutes les tonies. Ainsi, la méthode proposée, incluant la décomposition en plusieurs notes et le calcul des niveaux de confiance, nécessite peu de temps de calcul, laissant ainsi suffisamment de temps de traitement pour créer des applications musicales telles qu'une application d'accompagnement automatique ou de professeur virtuel de musique.

11.7 Discussion

En se référant au tableau 11.4, même avec le rejet selon l'intensité sonore et les différents rejets selon les niveaux de confiance, la quantité de notes estimées en trop demeure importante. Cette quantité est due au fait que quatre tonies ont été estimées pour chacune des trames, et ce, même lorsqu'une trame contient qu'une seule tonie. Certes, cette quantité peut être considérablement réduite en utilisant un meilleur estimateur de tonies qui détecterait plus de tonies avec succès et qui donnerait moins de tonies en trop. Cependant, il ne faut pas perdre de vue que cette couche a pour objectif d'estimer des notes en présence de ce type d'imperfection comme la réalisation d'un estimateur de tonies parfait est utopique.

Notez que de ces 52,0% de notes estimées en trop, 13,5% sont des notes en erreur d'octave, 31,5% sont des notes ayant une position de début ou de fin incorrecte et seulement 7% des cas sont des notes dont les tonies sont estimées en trop, et ce, sans rapport avec les notes jouées. Par conséquent, en améliorant le niveau de confiance de début ou de fin, il serait possible de réduire significativement le nombre de notes estimées en trop.

Une application cliente, comme une application d'accompagnement automatique, possède une information supplémentaire à la couche de transcription automatique pour identifier les notes estimées en erreur, soit la partition de musique. En effet, les notes estimées doivent être cohérentes avec les notes contenues dans la partition de musique. Si l'objectif de l'application est simplement de suivre un musicien à travers la partition de musique, la plupart des notes en trop pourraient facilement être ignorées advenant que l'application arrive à suivre le musicien avec les notes correctement estimées. En effet, la plupart des notes sont estimées correctement, mais avec une mauvaise position de début ou de fin, c'est-à-dire 31,5% pour table 11.4. D'ailleurs, les algorithmes d'alignement devraient, dans une certaine mesure, résister aux notes manquantes et aux notes estimées en erreur.

CHAPITRE 12

PERFORMANCE : COUCHE D'ALIGNEMENT

Ce chapitre présente les résultats de performance obtenus pour la couche d'alignement présentée sommairement au chapitre 5 et en détail au chapitre 8, soit la couche responsable d'aligner les notes estimées par la couche de transcription automatique aux notes de la partition jouée par le musicien. Essentiellement, ce chapitre décrit la performance sur les plans suivants :

1. Performance en présence de partitions monophoniques de musique ;
2. Performance en présence de partitions polyphoniques de musique ;
3. Performance pour des segments joués de façon isolés ;
4. Performance de la fenêtre de calcul définie à la section 8.2.1 du chapitre 8.

12.1 Performance - partitions monophoniques

La couche d'alignement de ce projet a été conçue pour fonctionner avec des partitions polyphoniques de musique, et ce, tout en conservant l'avantage principal du système d'alignement monophonique développé lors de la maîtrise [22], c'est-à-dire de permettre de retrouver le musicien dans la partition, et ce, sans même savoir à priori ce que le musicien jouera pourvu que ce dernier joue un segment suffisamment long pour que la couche d'alignement puisse retrouver le segment en question.

Bien qu'elle ait été conçue pour fonctionner en polyphonie, la couche d'alignement peut évidemment fonctionner avec des partitions monophoniques de musique. Cette première section a donc pour objectif de comparer la couche d'alignement avec l'algorithme monophonique FAMS (*Fuzzy Alignment of Musical Score*), soit l'algorithme développé lors de la maîtrise [22].

Afin d'évaluer la performance de la couche d'alignement en monophonie, le thème et les quatre premières variations du deuxième mouvement de la sonate II pour violon de Prokofiev joué par Gil Shaham [59] ont été utilisés. Rappelons que le tableau 6.2 du chapitre 6 résume les différentes parties de ce deuxième mouvement. Cette performance musicale a essentiellement été sélectionnée pour les raisons suivantes :

1. L'enregistrement audio est facilement accessible par Internet.

2. Il s'agit d'un enregistrement professionnel.
3. Le violon est joué en solo (sans accompagnement).
4. Elle contient des segments joués avec des vitesses variées.
5. Elle contient des notes jouées avec des types d'attaques variés.
6. Elle est composée de six parties dont les cinq premières sont monophoniques.
7. La pièce comporte plusieurs parties ayant des niveaux de difficulté variés (amateur à professionnel).

Pour l'évaluation, le nombre d'événements alignés correctement, inadéquatement et non alignés a été calculé pour chacune des parties du deuxième mouvement de la sonate où un événement correspond à un moment dans le temps où une ou plusieurs notes sont jouées en simultanées. Par exemple, pour le cas de la figure 8.3 du chapitre 8 présentée à la page 107, le nombre d'événements équivaldrait à 4.

Le tableau 12.1 présente les résultats obtenus avec l'algorithme FAMS et le système proposé, soit la sortie de la couche d'alignement. Pour ces résultats, un événement est considéré comme adéquatement aligné lorsqu'il y a une erreur inférieure à 0.5 seconde entre son moment de début réel et son moment de début obtenu par l'algorithme d'alignement concerné.

Tableau 12.1 Performance de la couche d'alignement pour des partitions monophoniques

Partie évaluée	Système d'alignement	Nombre d'événements					
		Adéquatement alignés		Inadéquatement alignés		Non alignés	
		Nb.	%	Nb.	%	Nb.	%
Tema	FAMS	40	100.0	0	0.0	0	0.0
	Système proposé	40	100.0	0	0.0	0	0.0
Variation I	FAMS	78	94.0	1	1.2	4	4.8
	Système proposé	77	92.8	0	0.0	6	7.2
Variation II	FAMS	62	57.4	7	6.5	39	36.1
	Système proposé	58	53.7	3	2.8	47	43.5
Variation III	FAMS	39	86.7	0	0.0	6	13.3
	Système proposé	41	91.1	0	0.0	4	8.9
Variation IV	FAMS	116	66.3	1	0.6	58	33.1
	Système proposé	121	69.1	0	0.0	54	30.9
Tema + Var. I à IV	FAMS	309	68.5	27	6.0	115	25.5
	Système proposé	354	78.5	6	1.3	91	20.2

Les parties intitulées *Tema*, *Var I* et *Var III* de la sonate pourraient être jouées par des violonistes amateurs puisque les notes sont jouées lentement. C'est d'ailleurs ces parties qui donnent les meilleurs résultats, et ce, tant pour l'algorithme FAMS que pour le système

proposé. Pour les parties *Tema* et *Var I*, on remarque des performances équivalentes pour les deux méthodes d'alignement. Cependant, pour la partie *Var III*, le système proposé est légèrement supérieur à l'algorithme FAMS, soit 91.1% d'événements adéquatement alignés par rapport à 86.7% pour l'algorithme FAMS. On remarque aussi que, pour ces parties, le nombre d'événements inadéquatement alignés pour le système proposé est nul. Ceci est avantageux puisque des événements inadéquatement alignés peuvent avoir des effets nuisibles sur des applications de musique découlant de ces algorithmes. En effet, en présence de mauvais alignements, une application pourrait, par exemple, afficher des erreurs de performance que le musicien n'a pas réellement faites ou encore tourner la page à un mauvais moment. Pour leur part, les événements non alignés sont beaucoup moins néfastes, et ce, particulièrement s'ils sont peu nombreux et éparpillés à travers la partition de musique.

Pour ce qui est de la partie *Var II*, on remarque que, quel que soit l'algorithme, la performance n'est pas très élevée. Celle-ci est due au fait que cette partie contient des segments très répétitifs tels que le segment dans la figure 12.1, et ce, joués rapidement, environ 7 notes à la seconde, avec des durées de notes très courtes, c'est-à-dire avec l'articulation staccato qui signifie que la durée de la note doit être raccourcie et complétée avec du silence. De plus, au violon, les notes jouées rapidement avec l'articulation staccato ne sont pas toujours claires puisque l'archet sautille sur la corde.



Figure 12.1 Exemple de segment répétitif contenu dans la partie *Var II*

Pour la partie *Var IV*, on remarque une légère amélioration pour le système proposé par rapport à l'algorithme FAMS, soit 69.1% par rapport à 66.3%. Notons que pour cette partie les notes sont jouées très rapidement, soit environ 10 notes à la seconde pour la majeure partie du temps. D'ailleurs, cette vitesse est près de la limite maximale d'un violoniste professionnel.

Finalement, les deux dernières lignes du tableau 12.1 indiquent un gain important de performance avec le système proposé, soit 78.5% par rapport à 68.5% lorsque toutes les parties sont alignées ensemble. Dans ce cas, au lieu d'aligner qu'une partie isolée de la sonate, toutes les parties évaluées précédemment ont été alignées en une seule fois. En d'autres mots, le système proposé et l'algorithme FAMS avaient cette fois en entrée le signal audio composé de toutes les parties évaluées précédemment.

Ces résultats démontrent que malgré le passage à la polyphonie, le système d'alignement proposé est plus performant que son prédécesseur monophonique, soit l'algorithme FAMS. En fait, ceci s'explique, entre autres, par le fait qu'un signal de musique où les notes sont jouées une à la suite de l'autre, soit de façon monophonique peut sur le plan audio produire un signal polyphonique. Par exemple, pour le cas du violon composé de 4 cordes, une note jouée sur une corde pourrait continuer de vibrer avec plus d'intensité qu'une autre note jouée successivement sur une autre corde.

12.2 Performance - partitions polyphoniques

Le tableau 12.2 présente la performance du système d'alignement proposé en polyphonie. Pour ce cas, la pièce sélectionnée est le deuxième mouvement (Fuga) de la sonate II pour violon de Bach en La mineur jouée par James Ehnes [1]. Il est à noter que la performance en polyphonie n'a pas été comparée à un autre système d'alignement puisque la performance des systèmes d'alignement présentés dans la littérature scientifique est décrite subjectivement, soit sans valeurs quantifiables. De plus, le niveau de détail fourni dans les articles scientifiques est insuffisant pour la reproduction exacte de ces systèmes d'alignements. En contrepartie, les résultats du tableau 12.2 pourraient servir comme base de comparaison pour les systèmes d'alignement futurs puisque la pièce utilisée pour évaluer la performance est disponible en ligne et le critère utilisé pour étiqueter une note comme correctement alignée est documenté dans la section 12.1.

Comparativement à l'évaluation de la performance en monophonie, une seule partie a été utilisée pour évaluer la performance du système en polyphonie. Néanmoins, cette partie contient près de 4 fois le nombre d'événements contenu dans l'ensemble de toutes les parties évaluées dans le tableau 12.1. De plus, au violon, la partie sélectionnée dans le tableau 12.2 peut être considérée comme hautement polyphonique, très difficile à jouer et, par conséquent, de calibre professionnel.

Tableau 12.2 Performance de la couche d'alignement pour des partitions polyphoniques

Partie évaluée	Système d'alignement	Nombre d'événements					
		Adéquatement alignés		Inadéquatement alignés		Non alignés	
		Nb.	%	Nb.	%	Nb.	%
Fuga	Système proposé	1549	91.2	3	0.2	147	8.7

Bien que la performance en polyphonie soit nettement supérieure à celle en monophonie, soit 91.2% comparativement à 78.5%, il faut comprendre que la performance dépend des pièces de musique sélectionnées. Au violon, comme il est très difficile de jouer plus d'une

note en simultanée, les segments polyphoniques sont joués lentement. Conséquemment, les notes de ces segments sont généralement suffisamment longues pour permettent leur juste détection par le système proposé telle que c'est le cas pour la pièce de musique sélectionnée. En addition, cette pièce ne contient pas de section répétitive telle que pour le cas illustré à la figure 12.1.

En plus d'être relativement performant en polyphonie, le système proposé est assez juste puisque seulement 0.2% des événements ont été alignés inadéquatement. D'ailleurs, en analysant cette performance, on peut constater qu'il y a encore place à l'amélioration, et ce, notamment en réduisant davantage le nombre d'événements non alignés.

12.3 Performance pour des segments joués de façon isolés

Cette sous-section a pour objectif de déterminer la performance du système lorsqu'il doit suivre un musicien pratiquant une pièce de musique, et ce, particulièrement lorsque ce dernier joue aléatoirement des segments courts composés de quelques mesures seulement. Pour ce faire, 34 segments isolés d'une longueur de 30 événements ont été sélectionnés à partir de l'enregistrement audio polyphonique discuté à la section 12.2, soit la Fuga de Bach jouée par James Ehnes [1]. Rappelons qu'un événement correspond à un moment dans le temps où une ou plusieurs notes sont jouées en simultanée. Le premier événement de chacun des segments a été choisi à tous les multiples de 50, et ce, jusqu'à la fin de la partition de musique contenant un total de 1699 événements.

Le tableau 12.3 présente les résultats obtenus pour cette évaluation. Il est à noter que pour chacune des lignes de résultat, le système a dû chercher le segment joué à travers toute la partition de musique. De plus, dès le début de l'analyse du segment, le programme d'évaluation a exécuté des requêtes au système pour obtenir l'alignement estimé, et ce, environ une fois par seconde. La performance a été évaluée à partir des quatre critères suivants :

1. ***Nombre de mauvais alignements avant le bon alignement***

Nombre d'alignements en erreur, soit un alignement qui ne correspond pas à ce qui a été réellement joué, et ce, avant que le bon alignement soit estimé par le système ;

2. ***Temps écoulé en secondes avant le bon alignement***

Temps écoulé en secondes avant que le système estime correctement l'alignement ;

3. ***Nombre d'événements joués avant le bon alignement***

Nombre d'événements joués avant que le système estime correctement l'alignement ;

4. Nombre d'événements non alignés au début du segment

Nombre d'événements au début que le système n'a pas été en mesure d'aligner dans son alignement.

Tableau 12.3 Performance de l'algorithme d'alignement lorsque des segments de 30 notes de la partition sont joués de façon isolée

No. du segment	Segment de 30 notes joués	Nb. mauvais alignements avant le bon alignement	Temps écoulé en secondes avant le bon alignement	Nb. d'événements joués avant le bon alignement	Nb. d'événements non alignés au début du segment
1	1-30	1	12.3	34	0
2	51-80	1	–	–	–
3	101-130	1	7.7	23	1
4	151-180	0	4.1	14	1
5	201-230	0	4.1	11	1
6	251-280	0	4.0	19	1
7	301-330	0	–	–	–
8	351-380	0	5.8	21	6
9	401-430	0	4.2	16	1
10	451-480	2	8.2	23	0
11	501-530	3	–	–	–
12	551-580	0	2.3	11	0
13	601-630	0	4.0	13	2
14	651-680	1	–	–	–
15	701-730	0	4.3	22	3
16	751-780	0	8.4	33	0
17	801-830	0	8.0	19	6
18	851-880	1	6.1	15	0
19	901-930	1	8.4	23	0
20	951-980	0	6.2	13	1
21	1001-1030	1	–	–	–
22	1051-1080	0	2.1	10	0
23	1101-1130	0	4.2	17	8
24	1151-1180	0	4.0	19	1
25	1201-1230	0	6.3	19	1
26	1251-1280	0	6.3	32	3
27	1301-1330	0	2.1	10	0
28	1351-1380	0	4.1	10	1
29	1401-1430	0	4.2	12	1
30	1451-1480	0	4.1	12	1
31	1501-1530	0	4.1	11	1
32	1551-1580	0	4.0	12	0
33	1601-1630	0	2.0	5	0
34	1651-1680	0	4.2	8	0
	<i>Moyenne</i>	0.4	5.2	16.8	1.4
	<i>Écart-type</i>	0.7	2.3	7.4	2.0
	<i>Médiane</i>	0	4.2	15.0	1
	<i>Minimum</i>	0	2.0	5	0
	<i>Maximum</i>	3	12.3	34	8

Selon ces résultats, le système estime adéquatement son premier alignement les trois quarts du temps environ, et ce, lorsqu'un alignement est demandé à chacune des secondes écoulées. Ainsi, il est préférable pour un programme client au système d'alignement d'attendre quelques secondes avant d'effectuer une requête d'alignement, et ce, pour éviter d'obtenir un alignement en erreur. D'ailleurs, pour cette pièce de musique, le système prend en

moyenne 5.2 secondes pour estimer correctement ce qui a été joué par le musicien, soit 2.0 secondes pour le meilleur temps d’alignement et 12.3 secondes pour le pire.

Comme le nombre de notes par seconde dépend de la pièce de musique, il pourrait être préférable pour un programme client d’attendre un certain nombre d’événements avant d’utiliser un alignement estimé par le système. Selon ces résultats, le système réussit à estimer correctement un alignement à l’aide d’environ 15 événements. Il est à noter que sur les 34 segments évalués, 5 segments auraient nécessité plus de 30 événements pour être alignés correctement par le système, c’est-à-dire les segments 2, 7, 11, 14 et 21.

12.4 Performance de la fenêtre adaptative de calcul

Cette section démontre l’importance de la fenêtre adaptative de calcul présentée à la sous-section 8.2.1 pour des pièces de musique de longue durée. Essentiellement, le temps nécessaire pour aligner la Fuga de Bach [1] d’une durée de 8.5 minutes a été chronométré avec et sans la fenêtre adaptative de calcul, et ce, en variant le nombre de cœurs du processeur à 1, 2 et 4 respectivement. Les résultats obtenus sont présentés dans le tableau 12.4.

Il est à noter que le système complet a été programmé en C++ avec des concepts informatiques favorisant le fonctionnement en temps réel tels que l’utilisation de fils multiples (*multithreading*), l’utilisation de la FFT (*Fast Fourier Transform*), l’utilisation de conteneurs variés en fonction du besoin et l’utilisation de pointeurs. De plus, la compilation des résultats a été réalisée à l’aide de l’ordinateur portable G53S de Asus muni d’une unité centrale de traitement Intel i7-2630QM de 2.0GHz sous le système d’exploitation Windows 7.

Tableau 12.4 Résultats avec la fenêtre adaptative de calcul

Nombre de cœurs utilisé du processeur	Temps de traitement (Minutes:secondes)	
	Sans fenêtre	Avec fenêtre
1	39:29	5:05
2	10:58	3:58
4	3:44	3:45

Ainsi, à partir des résultats du tableau 12.4, on peut remarquer que sans la fenêtre adaptative de calcul, le processeur de l’ordinateur utilisé pour réaliser l’alignement devrait être muni minimalement de 4 cœurs (*quad-core*) pour pouvoir aligner cette pièce en temps réel puisque, pour le cas d’un et deux cœurs, le temps nécessaire pour aligner cette pièce en différé est supérieur à la durée totale de la pièce, soit 8.5 minutes. Par contre, en présence

de la fenêtre adaptative de calcul, on remarque qu'un seul cœur suffirait pour aligner la pièce en temps réel.

12.5 Discussion

La performance de la couche d'alignement permet de répondre à une partie de la question de recherche, soit la partie surlignée ci-dessous :

Est-ce qu'il est possible pour un système informatique de suivre de façon autonome, en temps réel et sans entraînement une prestation musicale de niveau avancée, c'est-à-dire composée de notes rapides et polyphoniques et, ensuite, d'identifier les erreurs par rapport à la partition de musique jouée ?

Selon les résultats présentés dans le tableau 12.3, il est possible pour un système informatique de suivre de façon autonome et sans entraînement une partition de musique de niveau avancé contenant des notes polyphoniques comme le système développé n'a pas besoin d'être entraîné pour retrouver un musicien qui joue des segments de la Fuga de Bach de façon isolés, soit une pièce de violon de niveau très avancé contenant plusieurs notes jouées en polyphonie. En effet, pour ce cas, le système prend en moyenne 5.2 secondes pour retrouver le musicien à travers la partition. Il est à noter que le système développé pourrait être amélioré davantage comme il arrive à retrouver le musicien que dans 85% des cas.

Cependant, selon les résultats présentés dans le tableau 12.1, le système développé a tout de même de la difficulté à suivre une partition de musique contenant des notes jouées très rapidement. En effet, seuls 53.7% et 69.1% des notes ont été alignées avec leur note respective de la partition de musique, et ce, pour les variations II et IV respectivement, soit les variations les plus rapides du deuxième mouvement de la sonate II pour violon de Prokofiev. Heureusement, pour ces deux variations, le système aligne inadéquatement que 2.8% des notes pour le cas de la variation II et n'aligne aucune note inadéquatement pour le cas de la variation IV.

Comme le système développé est encore très jeune, il est raisonnable de croire qu'il est encore possible d'améliorer significativement sa performance en présence de notes rapides et, par conséquent, d'affirmer que la réponse à la partie surlignée de la question de recherche est positive. Cependant, à ce stade, la question peut être répondue que pour le cas du violon. En effet, certains instruments, tels que le piano, peuvent aussi jouer des notes rapides et, plus particulièrement, peuvent jouer beaucoup plus de notes en simultanées

que le violon comme le violon peut seulement jouer un maximum de quatre notes en simultané.

CHAPITRE 13

PERFORMANCE DE LA COUCHE COMPARATIVE

Ce chapitre présente les résultats de performance obtenus pour la couche comparative présentée sommairement au chapitre 5 et en détail au chapitre 9, soit la couche responsable d'estimer les erreurs de performance réalisées par le musicien, et ce, à partir de l'alignement estimé par la couche d'alignement et des informations contenues à l'intérieure de la partition de musique.

Pour l'évaluation, une démarche représentative d'une situation réelle d'usage des informations estimées par la couche comparative a été simulée. Pour ce faire, un professeur de violon a d'abord été recruté avec le mandat de sélectionner les pièces qui seront utilisées pour l'évaluation et, ensuite, de sélectionner deux violonistes de niveau intermédiaire qui, pour leur part, avaient le mandat de réaliser les enregistrements nécessaires à l'évaluation de cette couche. Il est à noter que cette démarche a été choisie pour éviter que l'expérimentateur impose un biais à ces derniers tests. Ceci dit, deux pièces monophoniques de niveau intermédiaire pour violon de Cristina Mondiriu ont été sélectionnées par le professeur, c'est-à-dire les pièces suivantes :

1. Hongrie - Viorica avec son violon en Europe de l'Est
2. Andantino en sol majeur - Caprices pour les jeunes violonistes

L'objectif de cibler des violonistes de niveau intermédiaire pour réaliser les enregistrements était pour assurer la présence d'erreurs de performance. D'ailleurs, tel que prévu, les deux pièces ont été jouées modestement par les deux violonistes et, par conséquent, les performances musicales enregistrées avaient plusieurs erreurs de performance.

Le tableau 13.1 présente la compilation des résultats obtenus pour ces deux pièces, et ce, pour les trois types d'erreur énumérés ci-dessous. Il est à noter que les détails de chacune des erreurs de performance estimées par la couche comparative sont fournis à l'annexe E.

1. **Début** : Erreur sur le moment de l'attaque (section 9.4)
2. **Durée** : Erreur sur la durée (section 9.5)
3. **Justesse** : Justesse de l'intonation (section 9.6)

Tableau 13.1 Compilation des résultats de performance de la couche comparative

	Type d'erreur		
	Début	Durée	Justesse
D'accord	34%	28%	83%
Désaccord	63%	34%	7%
Neutre	3%	38%	10%

Plus précisément, chacune des erreurs de performances détectées par la couche ont été validée à posteriori, et ce, en réalisant plusieurs écoutes des enregistrements et en notant chacune des erreurs selon les trois critères suivants :

1. **D'accord** : L'erreur est apparente et significative par rapport à l'ensemble de l'interprétation.
2. **Désaccord** : L'erreur est peu apparente et non significative par rapport à l'ensemble de l'interprétation.
3. **Neutre** : La caractérisation de l'erreur est difficile compte tenu des circonstances l'entourant.

Ces résultats dévoilent l'incapacité de la couche comparative à détecter adéquatement les erreurs de performance de type *début* et *durée*, et ce, malgré toutes les stratégies citées au chapitre 9. Bien entendu, ces résultats s'expliquent par le fait que la couche comparative est la dernière couche de traitement et, par le fait même, écope des imperfections des couches inférieures. Il ne faut pas nécessairement conclure que les stratégies proposées au chapitre 9 sont à délaissier, mais plutôt que les couches inférieures doivent être perfectionnées davantage, et ce, particulièrement pour améliorer l'estimation des débuts et fins de notes. D'ailleurs, il est pertinent de rappeler que la couche d'alignement peut aligner les notes malgré la présence de ces imperfections.

On remarque un pourcentage significatif d'erreurs catégorisées comme *neutre* pour le type *durée*, soit environ 38%. Ceci s'explique par le fait qu'un nombre important de notes sont jouées sur des cordes vides, c'est-à-dire jouées sans la présence d'un doigt sur la touche du violon. Le problème avec ce type de note est que ces dernières résonnent habituellement plus longtemps une fois que l'archet a cessé de les froter, et ce, comparativement à des notes où un doigt serre la corde. En effet, comparativement à une note jouée sur une corde vide, une fois jouée, le violoniste retire son doigt ayant pour effet d'arrêter le régime permanent de vibration de la corde et, par conséquent, de terminer rapidement la résonance de la note. Il est à noter que des conditions pourraient être ajoutées à la

couche comparative pour ne pas tenir compte des notes jouées trop longtemps lorsque ces dernières correspondent à l'une des quatre cordes du violon. Cependant, ces conditions seraient propres au violon et la couche comparative décrite au chapitre 9 a été conçue pour s'adapter à n'importe quel instrument.

Pour terminer, les résultats sont beaucoup plus encourageants pour les erreurs de type *justesse* puisqu'environ 83% de ces erreurs sont apparentes et significatives. Ceci s'explique par le fait que même si une partie de la note est manquante, soit au début ou à la fin, l'erreur sur sa justesse peut tout de même être perceptible. Ainsi, les imperfections sur l'estimation des débuts et fins de notes sont moins néfastes sur ce type d'erreur.

13.1 Discussion

La performance de la couche comparative permet de répondre à une partie de la question de recherche, soit, plus spécifiquement, les fragments surlignés ci-dessous :

Est-ce qu'il est possible pour un système informatique de suivre de façon autonome, en temps réel et sans entraînement une prestation musicale de niveau avancée, c'est-à-dire composée de notes rapides et polyphoniques et, ensuite, d'identifier les erreurs par rapport à la partition de musique jouée?

Selon les résultats obtenus, seules les erreurs de justesse sur l'intonation des notes sont estimées adéquatement, soit environ 83%. Par conséquent, à partir de ces résultats, il n'est pas possible d'affirmer clairement qu'il est possible pour un système informatique d'identifier automatiquement tous les types d'erreur de performance musicale. Néanmoins, ces résultats démontrent qu'il est beaucoup plus ambitieux de tenter d'estimer des erreurs de performance que de suivre un musicien à travers une partition de musique comme la performance de la couche d'alignement est nettement supérieure à celle de cette couche.

CHAPITRE 14

CONCLUSION

14.1 Sommaire des travaux réalisés

Cette thèse propose un système informatique permettant de créer des applications musicales fonctionnant en temps réel de type meilleur effort. Le cœur du système proposé est composé de quatre couches ayant chacune des fonctions bien distinctes, soit la couche de traitement du signal numérique, la couche de transcription automatique, la couche d'alignement et la couche comparative. Bien que la couche de traitement du signal ait été conçue spécifiquement pour le violon, les trois autres couches ont été conçues pour être indépendantes de l'instrument de musique.

La couche de traitement du signal a pour fonction d'estimer les tonies contenues dans un signal audio numérique correspondant à une pièce polyphonique quelconque jouée par un musicien, soit une pièce où plusieurs notes peuvent être jouées en simultanées. Pour chacune des valeurs de tonie estimée, la couche de traitement du signal leur associe deux valeurs de confiance, et ce, pour aider la couche supérieure de transcription automatique à estimer les tonies formant les notes de musique jouées par le musicien.

En effet, la couche de transcription automatique a pour objectif d'estimer les notes jouées par le musicien en fonction des tonies provenant de la couche de traitement du signal. Pour se faire, la couche de transcription automatique observe la corrélation entre les tonies dans le but de les regrouper pour créer une liste de notes potentiellement jouées par le musicien. Tout comme pour la couche de traitement du signal, la couche de transcription automatique transmet pour chacune des notes estimées des valeurs de confiance. Par la suite, la liste de notes est récupérée par la couche supérieure d'alignement qui a pour fonction de sélectionner les notes qui reflètent le mieux ce que le musicien a joué.

Comparativement à la couche de transcription automatique, la couche d'alignement utilise aussi comme entrée la partition de musique. Ainsi, elle compare les notes estimées par la couche inférieure avec les notes contenues dans la partition jouée par le musicien. Ces comparaisons permettent à la couche de situer le musicien dans la partition de musique. La sortie de cette couche est composée des notes estimées retenues et des notes de la partition correspondantes.

Pour terminer, la couche comparative récupère les notes de la couche inférieure pour estimer les déviations entre ce qui devait être joué selon la partition de musique et ce qui a réellement été joué. En fait, pour chacune des notes jouées, trois déviations sont calculées, soit la déviation sur le moment de début, sur la durée et sur la tonie.

Ces trois déviations peuvent finalement être récupérées par un module tel qu'une interface graphique, et ce, dans le but de présenter au musicien les erreurs qu'il a faites par rapport à ce qui était prescrit dans la partition de musique. Il est à noter que la sortie de la couche d'alignement peut aussi être utilisée pour créer des applications nécessitant de situer le jeu du musicien dans la partition de musique. Concrètement, le système informatique proposé dans cette thèse peut être compilé pour former une librairie utile pour réaliser diverses applications musicales telles qu'un professeur virtuel de musique ou une simple application permettant de tourner automatiquement les pages de la partition.

14.2 Sommaire des résultats obtenus

Les résultats obtenus pour la couche de traitement du signal démontrent que l'ensemble des algorithmes et stratégies utilisé pour cette couche permet d'obtenir 85.4% de tonies correctement estimées, et ce, pour 24.1% de tonies estimées en extra. De plus, en ajustant différemment les seuils de tolérance sur les niveaux de confiance, il est possible d'augmenter le pourcentage de tonies correctement estimées à 89.7%, et ce, au détriment d'un pourcentage supérieur de tonies estimées en extra, soit 44.4% pour ce cas particulier. Malgré le pourcentage élevé de tonies manquantes pour les deux cas énumérés ci-dessus, ces résultats s'avèrent tout de même supérieurs à ceux des algorithmes utilisés aux fins de comparaison.

Pour leurs parts, les résultats obtenus pour la couche de transcription automatique démontrent que le pourcentage de notes estimées correctement est près de 15% supérieur à celui obtenu à partir de la méthode typique, soit 85% par rapport à 70.2%. Il est à noter que la couche estime tout de même un pourcentage important de notes en extra, soit 28.5%. Cependant, la couche supérieure d'alignement a été conçue pour fonctionner adéquatement malgré un pourcentage élevé de notes estimées en extra. En fait, l'objectif de la couche de transcription automatique est d'estimer le maximum de notes correctement, et ce, malgré un nombre élevé inévitable de notes estimées en extra. D'ailleurs, la couche de transcription automatique associe, pour chacune des notes estimées, trois valeurs de confiance utilisées ensuite par la couche supérieure d'alignement. À titre indicatif, en conservant que les notes ayant des valeurs de confiance élevées, il est possible d'obtenir 80.6% de notes estimées correctement pour 52.0% de notes estimées en extra.

La couche d'alignement a, pour sa part, été comparée avec son prédécesseur, soit l'algorithme FAMS [22] conçu originalement pour des pièces monophoniques de musique. En fait, la couche d'alignement est une réforme importante de l'algorithme FAMS, et ce, pour l'adapter à des pièces polyphoniques de musique. Les résultats démontrent que, même pour des pièces monophoniques, la couche d'alignement améliore de 10% les résultats, soit 78.5% de notes correctement alignées par rapport à 68.5%. Pour la pièce de musique polyphonique Fuga [1] composée de plus de 1500 événements, soit un événement par regroupement de notes jouées en simultanées (accords), la couche permet d'obtenir 91.2% d'événements correctement alignés pour seulement 0.2% d'événements inadéquatement alignés.

Pour terminer, la couche comparative ne donne pas les résultats escomptés pour les déviations estimées sur les moments du début des notes et sur la durée des notes. Ceci s'explique par le fait que les débuts des notes estimées par la couche de transcription automatique ne sont pas suffisamment précis. Cependant, pour les déviations sur les tonies des notes, les résultats sont beaucoup plus encourageants puisque 83% des déviations retenues sont jugées comme apparentes et significatives.

14.3 Contributions

De toutes les couches développées dans le cadre de ce projet de recherche, la couche d'alignement apporte la plus importante contribution scientifique. En effet, comparativement aux systèmes d'alignements cités dans la littérature scientifique, cette couche permet de suivre en temps réel de type meilleur effort un musicien à travers une partition de musique, et ce, sans que le musicien ait besoin d'informer préalablement le système de l'emplacement exact qu'il jouera. Ainsi, la couche recherche d'abord l'emplacement joué à travers toute la partition de musique et, une fois trouvé, elle rétrécit progressivement sa fenêtre de recherche à travers la partition pour réduire la quantité de ressource CPU et mémoire nécessaire à l'alignement. Lorsque la qualité de l'alignement diminue, le système élargit sa fenêtre de recherche de nouveau pour retrouver l'emplacement joué par le musicien.

L'avantage de ce mode de fonctionnement est que même si le système est confus pour une période de temps indéterminée, il arrivera éventuellement à retrouver le musicien à travers la partition de musique, et ce, pourvu que le musicien joue un segment suffisamment long pour que le système puisse le situer. Ceci donne aussi la liberté au musicien de jouer les segments qu'il désire, et ce, sans jamais avoir à informer le système de ses intentions. Cette couche a aussi l'avantage de fonctionner pour des pièces polyphoniques de musique. Elle est aussi robuste en présence de performances musicales jouées par des musiciens

amateurs, c'est-à-dire des performances contenant plusieurs erreurs musicales et avec des tempos très variables.

Pour sa part, la couche de transcription automatique propose un système permettant de construire une liste de notes de musique estimées, et ce, à partir de tonies. La principale contribution de cette couche est qu'elle permet d'estimer des notes en présence de vibrato, de tonies manquantes et de notes dont la justesse varie en dehors des limites attendues.

Quant à elle, la couche de traitement du signal propose un estimateur de tonies optimisé spécifiquement pour le violon. Cet estimateur pourrait d'ailleurs être utilisé aux fins de comparaisons avec d'autres estimateurs spécialisés pour le violon comme ils sont peu nombreux dans la littérature scientifique.

14.4 Retour sur les objectifs fixés et la question de recherche

L'objectif principal du projet de recherche a été atteint puisque l'architecture logicielle et les algorithmes proposés permettent le développement d'applications musicales capables de suivre efficacement un musicien à travers une partition de musique tout en lui donnant la liberté de jouer la partition dans l'ordre qu'il le désire. Tel que souhaité, le tout fonctionne avec des partitions de musique complexes contenant des segments de notes rapides et des segments de notes jouées en polyphonie. Également, l'atteinte de cet objectif a permis de répondre à la première partie de la question de recherche présentée à l'introduction du document.

La couche de transcription automatique proposée permet d'atteindre l'un des deux objectifs complémentaires fixés, soit de proposer un estimateur de notes de musique où les notes seront estimées à partir de tonies estimées. Toutefois, le second objectif complémentaire n'a pas pu être atteint puisqu'il n'a pas été possible de démontrer le bien-fondé des stratégies proposées pour permettre la détection automatique d'erreurs de performance musicale telles que des notes jouées trop tôt ou trop tard, et ce, comme la couche de transcription automatique ne détecte pas convenablement les débuts des notes. Il est à noter que cette lacune a eu pour incidence de ne pas avoir de réponse pour la deuxième partie de la question de recherche.

14.5 Travaux futurs

La précision sur les débuts et fins de notes estimés par le système proposé pourrait être améliorée comme le manque de précision actuel affecte considérablement toutes les couches

du cœur du système, et ce, jusqu'à la dernière la couche, soit la couche comparative. En effet, cette dernière estime inadéquatement un nombre important de notes jouées trop tôt ou trop tard et de notes jouées trop ou pas assez longtemps. De plus, le pourcentage de notes estimées correctement augmenterait aussi comme certaines notes ont été détectées par le système, mais avec des débuts et fins de notes en dehors des plages acceptables pour être considérées comme correctement estimées. L'amélioration de la précision pourrait être réalisée, entre autres, par des algorithmes de traitement du signal audio dédiés à la détection d'événements transitoires. Il est à noter que ce problème serait probablement moins important si l'instrument de musique sélectionné pour la couche de traitement du signal produisait des débuts et fins de notes moins subtils que ceux du violon.

Le système éprouve aussi quelques difficultés en présence de segments de musique où un petit ensemble de notes est joué répétitivement tel que c'est le cas en présence d'un trille par exemple. Pour ces cas, le système devrait être perfectionné pour que la couche d'alignement effectue moins de comparaisons entre les notes estimées par le système et les notes de la partition, et ce, pour éviter que le système manque de ressources pour réaliser les comparaisons qui mèneraient à l'alignement optimal.

ANNEXE A

MODÈLES DE MARKOV CACHÉS

Cette annexe a pour objectif de compléter la section 3.1 du chapitre 3. Plus précisément, cette annexe explique en plus de détail le fonctionnement des modèles de Markov cachés (HMM, *Hidden Markov Models*). Les HMM sont utilisés avec succès dans plusieurs domaines tels que les systèmes de reconnaissances vocales et d'identification de locuteur. Comme illustré à la figure A.1, un HMM est composé de plusieurs états (points noirs) reliés entre eux par des flèches indiquant les transitions possibles d'un état à un autre. Les probabilités de transition d'un état à un autre sont inscrites au-dessus des flèches. De plus, chacun des états possède une distribution de probabilité d'un paramètre (symbole) observable du signal tel que l'énergie du signal. L'objectif principal du HMM est de trouver la séquence d'état la plus probable en observant les paramètres extraits du signal et les probabilités de transitions. Les différentes variables généralement utilisées par les HMM sont énumérées dans le tableau A.1.

Tableau A.1 Variables d'un HMM

Variable	Définition
S_j	État j
N	Nombre d'états du modèle
q_t	État au temps t
a_{ij}	Probabilité d'une transition de l'état i à l'état j
$b_j(k)$	Probabilité d'un symbole observable k à un état j
$A = \{a_{ij}\}$	Matrice des probabilités de transition a_{ij}
$V = \{v_1, v_2, \dots, v_M\}$	Symboles observables d'un état
M	Nombre de symboles observables par état
$B = \{b_j(k)\}$	Distribution de probabilité d'un symbole observable v_k à l'état j
$\pi = \{\pi_i\}$	Distribution de probabilité d'être initialement à l'état i
λ	Modèle spécifié par les paramètres A, B et π
$O = O_1, O_2, \dots, O_T$	Une séquence observée constituée de chacun des symboles observable v_k

Afin de mieux comprendre les HMM, la figure A.1 illustre un exemple simple utilisant un HMM pour reconnaître une note de piano contenu dans un signal audio. Ce HMM modélise une note de piano par trois états : attaque (S1), maintien (S2), relâchement (S3). Dans cet exemple, les paramètres observables associés à chacun des trois états précédents sont les suivants :

- S1 : $\Delta\text{Energie}$ (Différence d'énergie)
- S2 : f_0 (Fréquence fondamentale)
- S3 : Energie

Un exemple d'utilisation du HMM pour trouver la séquence d'état la plus probable à partir d'observations en utilisant la méthode directe. Bien que la méthode directe soit rarement

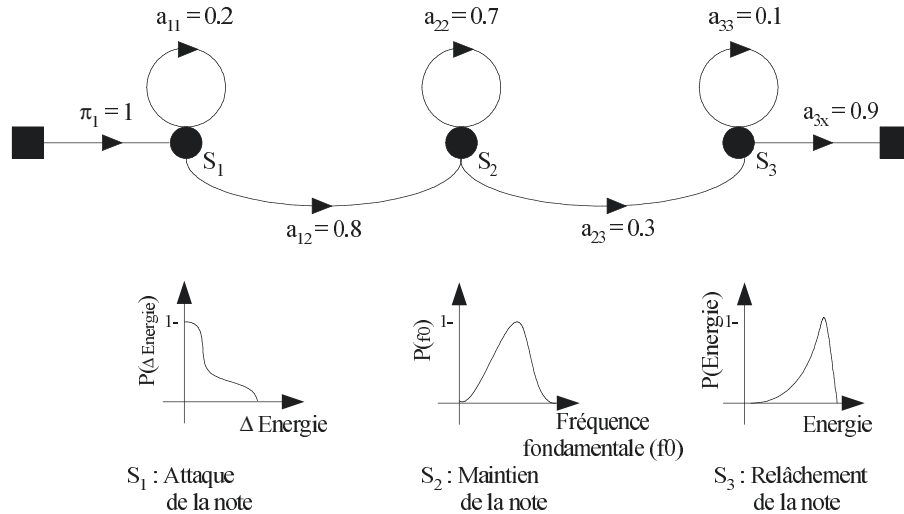


Figure A.1 Exemple d'un HMM utilisé pour reconnaître une note de musique

utilisée, elle est simple à comprendre et ainsi permettra de comprendre globalement le fonctionnement du HMM.

Question : En utilisant le modèle de la figure A.1 et les densités de probabilité de la figure A.2, quelle est la séquence d'états la plus probable sachant que la séquence d'observations $\mathbf{O} = O_1, O_2, O_3, O_4$ du tableau A.2 s'est produite¹ ?

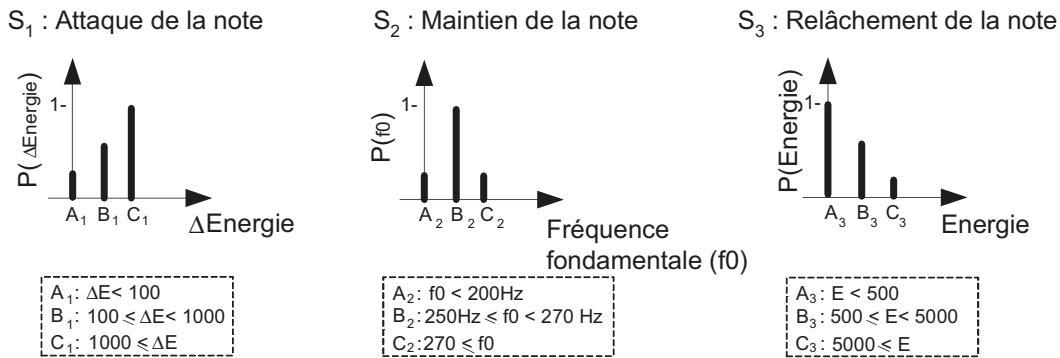


Figure A.2 Densités de probabilités pour le HMM de la figure A.1

Réponse :

Étape 1 Sélection d'une séquence de départ

Selon la figure A.1, il y a trois états ($N = 3$) et, selon l'énoncé du problème, la longueur de la séquence d'observation est de 4 ($T = 4$). Ainsi, il y a $N^T = 3^4 = 81$

1. Il est à noter que pour cet exemple, les valeurs A_x , B_x et C_x du tableau A.2 ont été obtenues à partir de cinq trames d'échantillons audio où O_1 est associée à la deuxième trame, O_2 à la troisième trame, etc. Selon la figure A.2, la valeur de C_1 de O_1 indique que la différence d'énergie ΔE entre la première et deuxième trame était supérieure à 1000. Pour sa part, la valeur C_2 de O_1 indique que la fréquence fondamentale retrouvée dans la deuxième trame était supérieure à 270Hz. Finalement, la valeur C_3 de O_1 indique que l'énergie de la deuxième trame était supérieure à 5000.

Tableau A.2 Séquence d'observations du problème pour l'exemple sur le DTW

O	O_1	O_2	O_3	O_4
ΔE	C_1	B_1	A_1	A_1
f_0	C_2	B_2	B_2	A_2
E	C_3	B_3	B_3	B_3

possibilités pour une séquence de départ. La séquence de départ peut-être n'importe lesquelles de ces possibilités.

Séquence de départ choisie (aléatoirement) : $Q = q_1, q_2, q_3, q_4 = S1, S2, S2, S3$

Étape 2 Calcul de la probabilité $P(Q|\lambda)$

$$P(Q|\lambda) = \pi_{q_1} \cdot a_{q_1q_2} \cdot a_{q_2q_3} \cdot a_{q_3q_4} = 1 \cdot 0.8 \cdot 0.7 \cdot 0.3 = 0.168$$

Étape 3 Calcul de la probabilité $P(O|Q, \lambda)$

$$P(O|Q, \lambda) = b_{q_1}(O_1) \cdot b_{q_2}(O_2) \cdot b_{q_3}(O_3) \cdot b_{q_4}(O_4)$$

En utilisant la figure A.2 et le tableau A.2 :

$$P(O|Q, \lambda) = b_{q_1}(C_1) \cdot b_{q_2}(B_2) \cdot b_{q_3}(B_2) \cdot b_{q_4}(B_3)$$

$$P(O|Q, \lambda) = 1 \cdot 1 \cdot 1 \cdot 0.5 = 0.5$$

Étape 4 Calcul de la probabilité conjointe $P(O, Q|\lambda)$

$$P(O, Q|\lambda) = P(O|Q, \lambda) \times P(Q|\lambda) = 0.168 \times 0.5 = 0.084$$

Étape 5 Choix d'une nouvelle séquence de départ Q_i

Étape 6 Retour à l'étape 2

...

Étape 7 Détenir la séquence la plus probable, donc la séquence donnant la plus grande valeur de $P(O, Q|\lambda)$.

L'essentiel à retenir est que la séquence la plus probable dépend des observations et des probabilités de transitions du HMM. En effet, le terme $P(O, Q|\lambda)$ représente la probabilité que la séquence sélectionnée Q et la séquence observée O se produisent simultanément sachant le modèle λ .

Pour plus de détail sur les HMM, consultez la référence [61].

ANNEXE B

DÉFORMATION DYNAMIQUE DU TEMPS

Cette annexe explique en plus de détail le fonctionnement de l'algorithme de la déformation du temps (DTW - *Dynamic Time Warping*). Le DTW est généralement utilisé pour aligner deux signaux de même nature, tels que deux signaux audio. Un exemple concret est présenté afin de faciliter la compréhension du DTW. Pour plus de détails, consultez la référence [33].

Le tableau B.1 représente deux interprétations de la gamme de Do où la fréquence fondamentale a été directement extraite. Le signal A représente une interprétation parfaite avec un rythme constant et des notes justes. Le signal B représente une interprétation de la gamme de Do contenant des irrégularités dans le rythme et des notes fausses.

Tableau B.1 Données pour l'exemple sur le DTW

No. Trame	1	2	3	4	5	6	7	8	9	10
Signal A (Hz.)	262	294	327	349	392	436	491	523	—	—
Signal B (Hz.)	262	262	294	330	349	349	392	440	494	523

La première étape pour aligner les deux signaux consiste à construire un tableau représentant les distances entre les paramètres extraits pour chacune des trames des signaux A et B. En utilisant les données du tableau B.1, on obtient les tableaux B.2 et B.3 où le tableau B.2 démontre le calcul effectué pour obtenir le tableau B.3, c'est-à-dire la valeur absolue de la différence entre les deux éléments comparés. Les valeurs calculées du tableau portent le nom de matrice de similarités. Il est à noter que la distance entre les paramètres extraits peut aussi être calculée par d'autres relations telles que la distance euclidienne.

L'alignement peut-être effectué en trouvant le trajet le plus court en distance en partant du point initial de la première colonne jusqu'au point final de la dernière colonne du tableau en avançant d'une case à la fois. L'algorithme DTW a pour fonction de trouver ce trajet. Dans le cas de notre exemple, la distance la plus courte correspond au trajet des valeurs encadrées du tableau B.3. Ainsi, la trame 1 du signal A correspond à la trame 1 du signal

Tableau B.2 Exemple d'un tableau utilisé pour les DTW pour démontrer les calculs

		Signal B									
		262	262	294	330	349	349	392	440	494	523
Signal A	262	262 - 262	262 - 262	262 - 294	262 - 330	262 - 349	262 - 349	262 - 392	262 - 440	262 - 494	262 - 523
	294	294 - 262	294 - 262	294 - 294	294 - 330	294 - 349	294 - 349	294 - 392	294 - 440	294 - 494	294 - 523
	327	327 - 262	327 - 262	327 - 294	327 - 330	327 - 349	327 - 349	327 - 392	327 - 440	327 - 494	327 - 523
	349	349 - 262	349 - 262	349 - 294	349 - 330	349 - 349	349 - 349	349 - 392	349 - 440	349 - 494	349 - 523
	392	392 - 262	392 - 262	392 - 294	392 - 330	392 - 349	392 - 349	392 - 392	392 - 440	392 - 494	392 - 523
	436	436 - 262	436 - 262	436 - 294	436 - 330	436 - 349	436 - 349	436 - 392	436 - 440	436 - 494	436 - 523
	491	491 - 262	491 - 262	491 - 294	491 - 330	491 - 349	491 - 349	491 - 392	491 - 440	491 - 494	491 - 523
	523	523 - 262	523 - 262	523 - 294	523 - 330	523 - 349	523 - 349	523 - 392	523 - 440	523 - 494	523 - 523
	0	0 - 262	0 - 262	0 - 294	0 - 330	0 - 349	0 - 349	0 - 392	0 - 440	0 - 494	0 - 523
	0	0 - 262	0 - 262	0 - 294	0 - 330	0 - 349	0 - 349	0 - 392	0 - 440	0 - 494	0 - 523

Tableau B.3 Matrice de similarité

		Signal B									
		262	262	294	330	349	349	392	440	494	523
Signal A	262	⓪	⓪	32	68	87	87	130	178	232	261
	294	32	32	⓪	36	55	55	98	146	200	229
	327	65	65	33	⓪	22	22	65	113	167	196
	349	87	87	55	19	⓪	⓪	43	91	145	174
	392	130	130	98	62	43	43	⓪	48	102	131
	436	174	174	142	106	87	87	44	⓪	58	87
	491	229	229	197	161	142	142	99	51	⓪	32
	523	261	261	229	193	174	174	131	83	29	⓪
	0	262	262	294	330	349	349	392	440	494	523
	0	262	262	294	330	349	349	392	440	494	523

B, la trame 2 du signal A correspond à la trame 1 du signal B, la trame 3 du signal A correspond à la trame 2 du signal B, etc. Généralement, le trajet optimal est trouvé à l'aide de technique de programmation dynamique et à l'aide d'algorithmes tels que l'algorithme de Viterbi [58].

Les méthodes d'alignement basées sur le DTW utilisent généralement des contraintes pour améliorer les résultats de l'alignement. Les contraintes indiquent les différents chemins que le trajet optimal peut utiliser pour avancer à travers la matrice de similarités. La figure B.1 présente deux contraintes utilisées par les chercheurs de la référence [76]. Comme certains chemins parmi les choix possibles peuvent être en général plus favorables, il est possible d'ajouter des poids (w_x) à chacune des façons d'avancer tels qu'illustrés à la figure B.1.

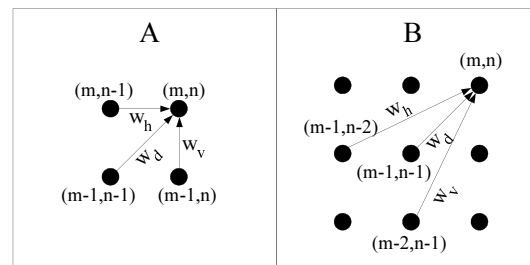


Figure B.1 Deux exemples de façon possible d'avancer à travers la matrice de similarités

Généralement, le choix des contraintes dépend des résultats obtenus expérimentalement et de la complexité à calculer le chemin optimal. Effectivement, une contrainte possédant 9 chemins possibles demandera beaucoup plus de traitement pour trouver le trajet optimal par rapport à une contrainte de 3 chemins.

ANNEXE C

VALEURS DE CONFIANCE D'UN CS

En lien avec le bloc d'alignement brut de la couche d'alignement présentée au chapitre 8, cette annexe présente les étapes de calculs nécessaires pour obtenir la valeur de confiance d'un nouvel élément ajouté à un chemin survivant (CS). Cette valeur de confiance reflète l'ensemble des attributs musicaux suivants :

1. Position de début attendue par rapport à celle obtenue pour l'élément ajouté
2. Durées attendues par rapport à celles obtenues pour les notes contenues dans l'élément ajouté
3. Consistance du tempo impliqué par l'élément ajouté par rapport à celui des éléments précédents
4. Proximité du tempo prescrit dans la partition par rapport à celui obtenu pour l'élément ajouté
5. Intensités des notes potentielles contenues dans le chemin survivant par rapport à celles qui ont été sautées

Afin d'aider à la compréhension des divers calculs, les deux axes fictifs des tableaux C.1 et C.2 sont utilisés à titre d'exemple. Il est à noter qu'à partir de ces axes, il est possible de construire la matrice de similarité de la figure C.1, et ce, en appliquant la méthode présentée à la sous-section 8.2.1.

Tableau C.1 Exemple d'axe d'index (j) d'un ensemble de notes à départ commun au niveau des notes détectées

	j=1	j=2	j=3	j=4	j=5	j=6	j=7	j=8	j=9
Identité	Ré4	Fa4	Sib4	Sol4	Fa4	La4	Ré4	La3	Ré4
Début (secondes)	0.340	0.946	1.352	1.761	2.443	2.240	2.656	2.888	3.137
Fin (secondes)	0.700	1.323	1.787	2.476	2.623	2.600	2.751	3.088	3.584
Intensité	14	139	161	130	118	27	18	212	184
Identité		Sib3	Ré4		Sib3		Mib4		Sib3
Début (secondes)	-	0.934	1.351	-	2.476	-	2.623	-	3.114
Fin (secondes)		1.307	1.795		2.619		3.137		3.557
Intensité		160	224		149		241		199
Identité		Sol4					Sol3		
Début(secondes)	-	1.260	-	-	-	-	2.641	-	-
Fin (secondes)		1.360					2.848		
Intensité		31					266		

Tableau C.2 Exemple d'axe d'index (i) d'un ensemble de notes à départ commun au niveau de la partition

	i=1	i=2	i=3	i=4	i=5	i=6	i=7
Identité	Fa4	Sib4	Sol4	Fa4	Mib4	La3	Ré4
Début (mesures)	1.000	1.250	1.500	1.875	2.000	2.125	2.250
Fin (mesures)	1.250	1.500	1.875	2.000	2.250	2.250	2.500
Tempo prescrit (secondes/mesures)	2.000	2.000	2.000	2.000	2.000	2.000	2.000
Identité	Sib3	Ré4	Do#4	Sib3	Sol3		Sib3
Début (mesures)	1.000	1.250	1.500	1.875	2.000		2.250
Fin (mesures)	1.250	1.500	1.875	2.000	2.125	-	2.500
Tempo prescrit (secondes/mesures)	2.000	2.000	2.000	2.000	2.000		2.000

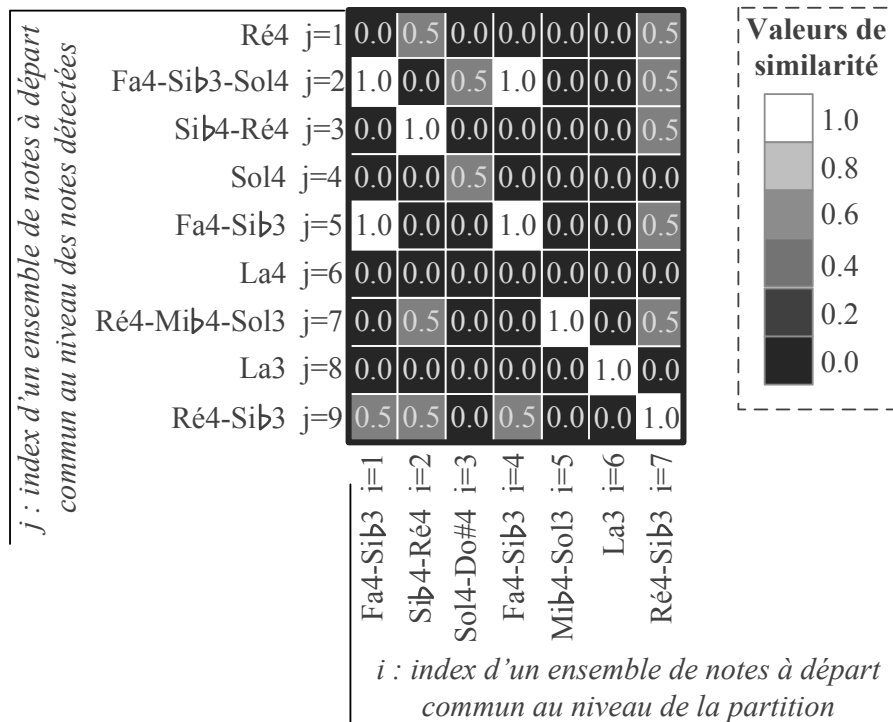


Figure C.1 Exemple fictif de matrice de similarité

C.1 Collecte des données pertinentes

Imaginons le chemin survivant de la figure C.2 composé de 7 éléments. La première étape consiste à recueillir l'information utile pour ce CS, et ce, à partir des informations contenues dans chacun des deux axes des tableaux C.1 et C.2 de la matrice de similarité de la figure C.1.

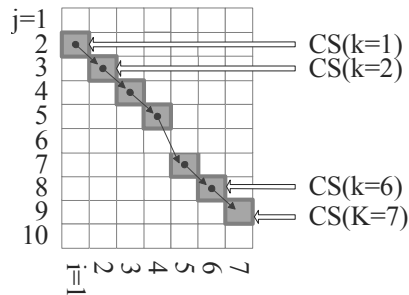


Figure C.2 Exemple de chemin survivant composé de 7 éléments

L'information à conserver correspond aux attributs de chacune des notes de même identité pour un élément donné k du CS. Ces informations sont regroupées dans le tableau C.3.

Tableau C.3 Information pertinente du CS de la figure C.2

	Coordonnées		Note number	Identité	Intensité	Début (secondes)	Médiane des débuts (secondes)	Fin (secondes)	Début (mesures)	Fin (mesures)	Tempo prescrit (secondes/mesure)
	i_k	j_k									
CS(k=1)	1	2	1	Fa4	139	0.946	0.940	1.323	1.000	1.250	2.000
		3	2	Sib3	160	0.934		1.307		1.250	
CS(k=2)	2	3	1	Sib4	161	1.352	1.352	1.787	1.250	1.500	2.000
			2	Ré4	224	1.351		1.795		1.500	
CS(k=3)	3	4	1	Sol4	130	1.761	1.761	2.476	1.500	1.875	2.000
CS(k=4)	4	5	1	Fa4	118	2.443	2.460	2.623	1.875	2.000	2.000
			2	Sib3	149	2.476		2.619		2.000	
CS(k=5)	5	7	1	Mib4	241	2.623	2.632	3.137	2.000	2.250	2.000
			2	Sol3	266	2.641		2.848		2.125	
CS(k=6)	6	8	1	La3	212	2.888	2.888	3.088	2.125	2.250	2.000
			2	Sib3	199	3.114		3.557		2.500	
CS(k=7)	7	9	1	Ré4	184	3.137	3.126	3.584	2.250	2.500	2.000

À partir de ces informations, il est maintenant possible d'effectuer les divers calculs nécessaires à l'obtention des valeurs de confiance de chacun des éléments d'un CS.

C.2 Calcul du tempo

Le tempo $\tau[k]$ en secondes par mesure d'un élément $k > 1$ est calculé selon l'équation C.1 où la variable $d_j[k]$ correspond à la médiane des valeurs de début des notes en secondes comprises dans l'élément k en question et $d_i[k]$ correspond au début en mesure de cet élément k .

$$\tau[k] = \max \left(0, \frac{d_j[k] - d_j[k-1]}{d_i[k] - d_i[k-1]} \right) \quad (\text{C.1})$$

Par exemple, le tempo du sixième élément du chemin survivant de la figure C.2 serait calculé comme suit :

$$\tau[6] = \max \left(0, \frac{2.888 - 2.632}{2.125 - 2.000} \right) = 2.048 \quad (\text{C.2})$$

En pratique, la partition de musique ne devrait jamais être jouée à reculons. Ainsi, pour tenir compte de cela, la fonction $\max()$ de l'équation C.1 est utilisée. De plus, pour obtenir la valeur $d_j[k]$, la médiane a été utilisée au lieu de la moyenne, et ce, dans le but de retirer les valeurs extrêmes pouvant être le résultat d'imperfections de la couche de transcription automatique.

C.3 Calcul du niveau de confiance relatif au moment de début

Le niveau de confiance relatif au moment de début d'un élément k est une mesure comprise dans la plage $[0,1]$ indiquant le degré de confiance d'un élément k , et ce, en comparant le début $d_j[k]$ mesuré du nouvel élément k et le début estimé $\hat{d}_j[k]$ en considérant les éléments précédents du chemin survivant. Cette valeur de confiance $c_1[k]$ se calcule à l'aide des équations C.3 et C.4.

$$c_1[k] = \begin{cases} 1.0, & k \leq 2 \\ \max(0.0, 1.0 - |d_j[k] - \hat{d}_j[k]|), & \text{Autrement} \end{cases} \quad (\text{C.3})$$

$$\hat{d}_j[k] = d_j[k-1] + \tau[k-1] (d_i[k] - d_i[k-1]) \quad (\text{C.4})$$

Par exemple, ce niveau de confiance pour le septième élément du chemin survivant de la figure C.2 serait calculé comme suit :

$$\hat{d}_j[7] = 2.888 + 2.048 (2.250 - 2.125) = 3.144 \quad (\text{C.5})$$

$$c_1[7] = \max(0.0, 1.0 - |3.126 - 3.144|) = 0.982 \quad (\text{C.6})$$

C.4 Calcul du niveau de confiance relatif aux durées

Le niveau de confiance relatif aux durées de notes d'un élément k est une mesure comprise dans la plage $[0,1]$ indiquant le degré de confiance d'un élément k , et ce, en comparant les durées de notes mesurées $\lambda_j[k, l]$ et les durées estimées $\hat{\lambda}_j[k, l]$ en considérant les éléments précédents du chemin survivant. Cette valeur de confiance $c_2[k]$ se calcule à l'aide des équations C.7, C.8 et C.9 où les variables $f_i[k, l]$ et $f_j[k, l]$ correspondent aux moments de fin d'une note l d'un élément k composé de L_k notes exprimés en mesures et en secondes respectivement.

$$c_2[k] = \begin{cases} 1.0, & k = 1 \\ \max\left(0, 1 - \frac{1}{L} \sum_{l=1}^{L_k} \max\left(0, \hat{\lambda}_j[k, l] - \lambda_j[k, l]\right)\right), & \text{Autrement} \end{cases} \quad (\text{C.7})$$

$$\hat{\lambda}_j[k, l] = \tau[k - 1] (f_i[k, l] - d_i[k]) \quad (\text{C.8})$$

$$\lambda_j[k, l] = f_j[k, l] - d_j[k, l] \quad (\text{C.9})$$

Par exemple, ce niveau de confiance pour le septième élément du chemin survivant de la figure C.2 serait calculé comme suit :

$$\hat{\lambda}_j[7, 1] = \hat{\lambda}_j[7, 2] = 2.048 (2.500 - 2.250) = 0.512 \quad (\text{C.10})$$

$$\lambda_j[7, 1] = 3.584 - 3.137 = 0.447 \quad (\text{C.11})$$

$$\lambda_j[7, 2] = 3.557 - 3.114 = 0.443 \quad (\text{C.12})$$

$$c_2[7] = \max\left(0, 1 - \frac{1}{2} \max(0, 0.512 - 0.447) - \frac{1}{2} \max(0, 0.512 - 0.443)\right) = 0.933 \quad (\text{C.13})$$

Il est à noter que la fonction $\max()$ à l'intérieur de la sommation permet de ne pas tenir compte des notes dont la durée mesurée est supérieure à la durée estimée. Cette décision tient compte du fait que la couche de transcription automatique détecte souvent les notes avec une durée plus longue que la durée réelle.

C.5 Calcul du niveau de confiance relatif à la consistance du tempo

Le niveau de confiance relatif à la consistance du tempo d'un élément k est une mesure comprise dans la plage $[0,1]$ indiquant le degré de confiance d'un élément k , et ce, en

comparant le tempo $\tau[k]$ impliqué par cet élément k par rapport au tempo de l'élément précédent $k - 1$. Cette valeur de confiance c_3 se calcule à l'aide de l'équation C.14.

$$c_3[k] = \begin{cases} 1.0, & k = 1 \\ \min(\tau[k-1], \tau[k]) / \max(\tau[k-1], \tau[k]), & \text{Autrement} \end{cases} \quad (\text{C.14})$$

Par exemple, ce niveau de confiance pour le septième élément du chemin survivant de la figure C.2 serait calculé comme suit :

$$c_3[7] = \frac{1.904}{2.048} = 0.9297 \quad (\text{C.15})$$

C.6 Calcul du niveau de confiance relatif à la proximité du tempo

Le niveau de confiance relatif à la proximité du tempo d'un élément k est une mesure comprise dans la plage $[0,1]$ indiquant le degré de confiance d'un élément k , et ce, en comparant le tempo mesuré $\tau[k]$ au tempo prescrit $\hat{\tau}[k]$ dans la partition de musique. Cette valeur de confiance $c_4[k]$ se calcule à l'aide de l'équation C.16.

$$c_4[k] = \begin{cases} 1.0, & k = 1 \\ \min(1.0, \min(\tau[k], \hat{\tau}[k]) / \max(\tau[k], \hat{\tau}[k]) + C), & \text{Autrement} \end{cases} \quad (\text{C.16})$$

La constante C de l'équation C.16 permet de donner un peu de liberté au musicien afin qu'il puisse jouer la pièce avec un tempo plus lent ou plus rapide que celui prescrit dans la partition. Lors de l'évaluation de la performance de la couche d'alignement, cette valeur a été ajustée à 0.25. Ainsi, le musicien peut jouer jusqu'à 25 % plus rapidement ou moins rapidement la partition de musique, et ce, tout en obtenant une valeur de confiance sur la proximité du tempo égale à 1 pour les éléments du chemin survivant approprié, soit la valeur de confiance la plus élevée.

Il est à noter que le tempo est généralement prescrit au début de la partition de musique. Il peut être exprimé par des mots italiens tels qu'andante, allegro et presto. Ces mots prescrivent des valeurs approximatives de battements par minute. Ainsi, il peut être nécessaire de convertir le tempo $\hat{\tau}[k]$ en secondes par mesure avant le calcul de l'équation C.16.

C.7 Calcul du niveau de confiance relatif à l'intensité des notes

Le niveau de confiance relatif à l'intensité des notes pour un élément k est une mesure comprise dans la plage $[0,1]$ comparant l'intensité perceptuelle $g_j[k, l]$ des notes faisant parties du chemin survivant par rapport à l'intensité $\tilde{g}_{j_k}[p]$ des notes ne faisant pas parties du chemin survivant. Cette valeur de confiance $c_5[k]$ se calcule à l'aide des équations C.17, C.18 et C.19. Pour l'exemple du tableau C.3, les intensités $g_j[k, l]$ correspondent aux éléments

de la cinquième colonne d'un élément k . Pour leur part, les éléments $\tilde{g}_{j_k}[p]$ correspondent aux intensités des notes du tableau C.1 ne faisant pas partis des éléments $k - 1$ et k du tableau C.3 à plus ou moins une octave près, et ce, pour les éléments du tableau C.1 dont les indices sont à l'intérieur de la plage $j_{k-1} \leq j \leq j_k$. Cette valeur de confiance s'appuie sur le fait que pour le cas d'un chemin survivant bien fondé, la somme des énergies des notes contenues dans l'élément k devrait être plus importante que celle des notes non utilisées de ce même élément et des éléments sautés de l'axe j .

$$c_5[k] = \begin{cases} 1.0, & k = 1 \\ \frac{\alpha[k]}{\alpha[k] + \beta[k]}, & \text{Autrement} \end{cases} \quad (\text{C.17})$$

$$\alpha[k] = \sum_{l=1}^{L_k} g_j[k, l] + \sum_{l=1}^{L_{(k-1)}} g_j[k - 1, l] \quad (\text{C.18})$$

$$\beta[k] = \sum_{p \in P_{j_k}} \tilde{g}_{j_k}[p] \quad (\text{C.19})$$

On remarque que certaines notes ne font pas parties de l'ensemble P_{j_k} de l'équation C.19, ceci s'explique par le fait que certaines des notes détectées par la couche de transcription automatique correspondent à des erreurs d'octave de notes existantes. Comme elles possèdent des harmoniques communes à des notes existantes, leur intensité est généralement non négligeable. De ce fait, les notes de l'ensemble P_{j_k} comprennent seulement les notes de l'axe $j_{k-1} \leq j \leq j_k$ qui ne sont pas de même identité à une octave près de l'une des notes contenues dans les éléments $k - 1$ et k du chemin survivant. Ainsi pour le chemin survivant de la figure C.2 lié aux données des tableaux C.1 à C.3, le niveau de confiance pour le cinquième élément serait calculé comme suit :

$$\alpha[5] = (241 + 266) + (118 + 149) = 774 \quad (\text{C.20})$$

$$\beta[5] = 0 + 27 + 18 = 45 \quad (\text{C.21})$$

$$c_5[5] = \frac{774}{774 + 45} = 0.945 \quad (\text{C.22})$$

C.8 Calcul du niveau de confiance global de l'élément

Le niveau de confiance global $c[k]$ d'un élément k , décrit par l'équation C.23, correspond simplement à la moyenne des cinq niveaux de confiances calculés lors des sections précédentes.

$$c[k] = \sum_{n=1}^5 c_n[k] \quad (\text{C.23})$$

C'est d'ailleurs ce niveau de confiance qui, entre autres, est utilisé pour déterminer si la construction d'un chemin survivant est poursuivie ou non, et ce, selon la première condition de la ligne 5 du pseudo-code de l'algorithme 1 présenté à la page 123.

ANNEXE D

PROCÉDURE POUR OBTENIR LA TONIE NOMINALE

En complément à la section 2.1 sur l'état de l'art des algorithmes d'estimation de tonies multiples, cette annexe précise la relation entre les notes de musique et le domaine des fréquences, et ce, afin de détailler le contexte mathématique permettant d'évaluer les tonies nominales des notes de musique.

La plupart des instruments modernes sont ajustés de façon à ce que toutes les notes adjacentes soient distancées entre elles par une distance constante de 100 cents (voir l'équation 2.5). En musique, cette façon de jouer est référée comme étant en *tempérament égal*. Ainsi, lorsqu'une pièce de musique est jouée en tempérament égal, il est possible de trouver la fréquence fondamentale f_i nominale d'une note à partir de l'équation D.1 où i correspond à l'index MIDI (Musical Instrument Digital Interface) de la note en question et f_{69} correspond à la fréquence de référence de la note dont l'index MIDI est égale à 69, c'est-à-dire la note de référence standard : La à 440Hz.

$$f_i = f_{69} \times 2^{\frac{i-69}{12}} \quad (\text{D.1})$$

Il est à noter que le tempérament égal n'est pas toujours respecté par les musiciens puisqu'il ne sonne pas tout à fait naturel à l'oreille. En fait, pour sonner naturel, la tonalité d'une note doit dépendre de la gamme sur laquelle une pièce est basée. En musique, cette façon de jouer est référée comme étant la gamme naturelle. Par exemple, en référence avec le tableau D.1, la tonie de la note *La* pour une pièce jouée en Sol majeur n'est pas la même que celle pour une pièce jouée en Do majeur. Il est à noter que la tonie nominale d'une note jouée dans une gamme naturelle se calcule à l'aide des ratios de la deuxième colonne du tableau D.1.

Néanmoins, l'avantage du tempérament égal est qu'un instrument accordé dans ce tempérament peut jouer n'importe quelles pièces de musique, et ce, peu importe la gamme de la pièce. Cet atout peut s'avérer un avantage important pour un instrument tel que le piano puisque ce dernier est généralement accordé que quelques fois par année. Par contre, pour d'autres instruments tels que la voix et le violon, le tempérament égal peut ne pas être respecté comme le musicien à plus de contrôle sur la tonie et, ainsi, peut aisément décider de jouer les notes selon la gamme naturelle, et ce, pourvu que les instruments qui l'accompagnent puissent aussi s'adapter à la gamme naturelle.

Il existe d'autres systèmes d'intervalle, mais les plus utilisés sont le tempérament égal et la gamme naturelle. Ainsi, la couche comparative ne traite que ces deux cas. Pour ce faire, la couche analyse par défaut les erreurs sur la tonie de chacune des notes jouées pour le cas du tempérament égal et, ensuite, pour le cas de la gamme naturelle. Une fois les deux

cas traités, la couche comparative utilise le système d'intervalle dont la somme cumulative de toutes les erreurs en cents est la plus faible. Il est à noter que le système d'intervalle peut aussi être sélectionné par le musicien à l'aide des paramètres de configuration, si ce dernier le désire.

Tableau D.1 Exemples de gammes naturelles et ratio permettant d'obtenir les fréquences fondamentales nominales des notes

Nom de l'intervalle	Rapport entre la note en question et la note fondamentale	Exemple pour la gamme de Do majeur		Exemple pour la gamme de Sol majeur	
		Note	Tonie (Hz)	Note	Tonie (Hz)
Fondamentale	1	Do	264.00	Sol	396.00
Demi-ton chromatique	25/24	Do♯	275.00	Sol♯	412.50
Demi-ton diatonique	16/15	Ré♭	281.60	Lab	422.40
Ton majeure	9/8	Ré	297.00	La	445.50
Seconde augmentée	75/64	Ré♯	309.38	La♯	464.06
Tierce mineure	6/5	Mib	316.80	Sib	475.20
Tierce majeure	5/4	Mi	330.00	Si	495.00
Tierce augmentée	125/96	Mi♯	343.75	Si♯	515.63
Quarte diminuée	32/25	Fab	337.92	Dob	506.88
Quarte	4/3	Fa	352.00	Do	528.00
Quarte augmenté	45/32	Fa♯	371.25	Do♯	556.88
Quinte diminuée	64/45	Solb	375.47	Ré♭	563.20
Quinte	3/2	Sol	396.00	Ré	594.00
Quinte augmentée	25/16	Sol♯	412.50	Ré♯	618.75
Sixte mineure	8/5	Lab	422.40	Mib	633.60
Sixte majeure	5/3	La	440.00	Mi	660.00
Sixte augmentée	225/128	La♯	464.06	Mi♯	696.09
Septième mineure	9/5	Sib	475.20	Fa	712.80
Septième majeure	15/8	Si	495.00	Fa♯	742.50
Septième augmentée	125/64	Si♯	515.63	Fa♯♯	773.44
Octave diminuée	48/25	Dob	506.88	Solb	760.32
Octave	2	Do	528.00	Sol	792.00

ANNEXE E

DÉTAILS DE PERFORMANCE DE LA COUCHE COMPARATIVE

Les tableaux E.1 et E.2 présentent le détail des résultats obtenus pour l'évaluation de la couche comparative, et ce, pour les pièces Hongrie et Andantino respectivement et telles que présentées au chapitre 13. Plus précisément, chacune des erreurs de performances détectées par la couche ont été validée à posteriori, et ce, en réalisant plusieurs écoutes des enregistrements et en notant chacune des erreurs selon les trois critères suivants :

1. **D'accord** : L'erreur est apparente et significative par rapport à l'ensemble de l'interprétation.
2. **Désaccord** : L'erreur est peu apparente et non significative par rapport à l'ensemble de l'interprétation.
3. **Neutre** : La caractérisation de l'erreur est difficile compte tenu des circonstances l'entourant.

Il est à noter que les erreurs sont triées par ordre décroissant selon leur valeur relative, et ce, pour chacun des types d'erreurs. L'erreur relative correspond à l'erreur divisée par la valeur médiane de toutes les erreurs d'un même type, et ce, pour l'ensemble de l'interprétation évaluée par la couche comparative.

La valeur médiane a été privilégiée par rapport à la valeur moyenne puisqu'elle permet d'éliminer les extremums, soit des erreurs aberrantes qui résultent généralement non pas du musicien, mais des imperfections du système en soi. Le nombre d'erreurs utilisé pour calculer la valeur médiane est égal au nombre de notes alignées puisqu'il est impossible pour un musicien d'arriver parfaitement sur la valeur cible, il existe donc une valeur d'erreur non nulle pour chacune des notes jouées. Quant à lui, le nombre total d'erreurs utilisé pour évaluer la performance de la couche comparative correspond à 10% du nombre total de notes alignées, et ce, pour chacune des interprétations. Il est à noter que pour les deux interprétations évaluées, l'erreur la moins significative utilisée pour l'évaluation de la couche possède une valeur relative non négligeable, soit 2.0.

Tableau E.1 Performance de la couche comparative avec l’Hongrie

Type d’erreur	No	Position (en mesure)	Erreur (en secondes)	Erreur relative (Erreur/Mediane)	Évaluation manuelle de l’erreur
Erreur sur le moment de l’attaque	1	33,00	6,2	88,8	Désaccord
	2	12,25	4,5	65,0	Désaccord
	3	14,25	-3,1	44,8	D’accord
	4	14,13	-2,9	41,5	D’accord
	5	22,88	2,5	36,2	Désaccord
	6	14,38	-1,0	15,0	D’accord
	7	37,19	-0,9	12,3	Désaccord
	8	34,38	-0,6	8,1	Désaccord
	9	17,13	-0,5	7,8	Désaccord
	10	24,38	-0,5	7,3	Désaccord
	11	37,13	-0,5	6,8	Désaccord
Erreur sur la durée	1	16,75	-0,5	15,0	Désaccord
	2	16,5	-0,4	12,3	Désaccord
	3	22,875	-0,3	10,0	Désaccord
	4	12,875	-0,3	8,3	Neutre
	5	33,75	0,3	7,3	D’accord
	6	12,375	0,2	7,0	D’accord
	7	3,25	0,2	6,8	D’accord
	8	12,25	0,2	6,8	D’accord
	9	34,25	-0,2	6,2	Désaccord
	10	3,875	0,2	6,0	D’accord
	11	3,375	0,2	5,5	D’accord
Erreur sur la tonie (cents)	1	17,13	-53,6	6,3	D’accord
	2	23,00	29,2	3,4	D’accord
	3	34,88	-24,9	2,9	D’accord
	4	14,75	-23,8	2,8	D’accord
	5	37,44	23,8	2,8	Neutre
	6	37,25	21,0	2,5	Neutre
	7	4,38	-20,5	2,4	D’accord
	8	24,00	20,2	2,4	D’accord
	9	15,75	-20,0	2,4	D’accord
	10	34,00	19,9	2,3	Neutre
	11	15,88	-19,8	2,3	D’accord

Tableau E.2 Performance de la couche comparative avec l'Andantino

Type d'erreur	No	Position (en mesure)	Erreur (en secondes)	Erreur relative (Erreur/Mediane)	Évaluation manuelle de l'erreur
Erreur sur le moment de l'attaque	1	15,81	-7,2	123,8	Désaccord
	2	22,13	-4,9	84,0	D'accord
	3	18,81	-4,1	70,8	D'accord
	4	3,88	-3,8	65,0	Désaccord
	5	23,06	3,1	54,2	Désaccord
	6	21,06	3,0	51,6	Neutre
	7	18,75	-1,8	31,0	D'accord
	8	23,19	-1,7	29,8	Désaccord
	9	15,44	-1,6	27,6	Désaccord
	10	13,69	-1,5	26,4	D'accord
	11	22,00	-1,2	20,0	D'accord
	12	22,06	-1,1	18,6	D'accord
	13	13,63	-1,0	17,2	D'accord
	14	13,38	-0,9	14,8	Désaccord
	15	13,25	-0,8	14,6	Désaccord
	16	5,88	-0,7	12,0	Désaccord
	17	6,00	-0,7	11,8	Désaccord
	18	8,00	-0,6	10,2	Désaccord
Erreur sur la durée	1	15,75	-1,3	20,5	Désaccord
	2	13,63	-0,5	8,5	D'accord
	3	3,63	-0,5	8,5	Désaccord
	4	23,38	-0,5	7,5	D'accord
	5	6,88	0,4	5,6	Neutre
	6	16,50	0,3	5,3	Neutre
	7	23,25	0,3	4,9	Désaccord
	8	6,63	0,3	4,2	Neutre
	9	8,38	0,3	4,2	Neutre
	10	6,38	0,2	3,6	Neutre
	11	8,63	0,2	3,5	Neutre
	12	3,00	0,2	3,3	Désaccord
	13	21,06	-0,2	3,3	Désaccord
	14	13,50	-0,2	3,1	Désaccord
	15	12,13	0,2	3,1	Neutre
	16	14,00	0,2	2,9	Neutre
	17	8,88	0,2	2,9	Neutre
	18	7,13	0,2	2,9	Neutre
Erreur sur la tonie (cents)	1	2,25	-31,7	3,8	D'accord
	2	13,56	-30,6	3,6	D'accord
	3	2,50	-29,4	3,5	D'accord
	4	13,38	27,6	3,3	D'accord
	5	10,00	-22,8	2,7	D'accord
	6	5,75	22,2	2,6	D'accord
	7	11,25	21,6	2,6	D'accord
	8	15,56	-21,3	2,5	Désaccord
	9	22,56	-20,9	2,5	D'accord
	10	10,25	-20,8	2,5	D'accord
	11	13,69	-20,0	2,4	D'accord
	12	11,50	18,7	2,2	D'accord
	13	14,25	17,9	2,1	D'accord
	14	11,75	17,4	2,1	Désaccord
	15	19,63	-17,4	2,1	D'accord
	16	10,13	-17,4	2,1	D'accord
	17	14,38	17,4	2,1	D'accord
	18	2,13	-16,7	2,0	D'accord

LISTE DES RÉFÉRENCES

- [1] J.S. Bach. Les six sonates et partitas pour violon seul ; violoniste : James Ehnes. Enregistrement audio, 2000.
- [2] M. A. Bartsch and G. H. Wakefield. To catch a chorus: using chroma-based representations for audio thumbnailing. In *IEEE Workshop on Application of Signal Processing to Audio and Acoustics*, pages 15–18, New Paltz, New York, October 2001.
- [3] J.P. Bello and M. Sandler. Phase-based note onset detection for music signals. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 441–444, Hong Kong, China, April 2003.
- [4] D.J. Benson. *Music A Mathematical Offering*. Cambridge University Press, 2007.
- [5] M. W. Buck. *The efficacy of SmartMusic (RTM) assessment as a teaching and learning tool*. PhD thesis, Ph. D diss., University of Southern Mississippi, 2008.
- [6] A. Camacho and J.G. Harris. A pitch estimation algorithm based on the smooth harmonic average peak-to-valley envelope. In *IEEE International Symposium on Circuits and Systems ISCAS*, pages 3940–3943, New Orleans, LA, June 2007.
- [7] P. Cano, J. Bonada, M. Boer, and X. Serra. Voice morphing system for impersonating in karaoke applications. In *International Computer Music Conference (ICMC)*, Berlin, Germany, 2000.
- [8] P. Cano, A. Loscos, and J. Bonada. Score-performance matching using HMMs. In *International Computer Music Conference (ICMC)*, pages 441–444, Beijing, China, 1999.
- [9] A. Cheveigné and H. Kawahara. Yin, a fundamental frequency estimation for speech and music. *Acoustical Society of America*, 111(4) :1917–1930, 2002.
- [10] C. d’Alessandro and M. Castellengo. The pitch of short-duration vibrato tones. *Journal of the Acoustical Society of America*, 95(3) :1617–1630, 1994.
- [11] R. B. Dannenberg. An on-line algorithm for real-time accompaniment. In *International Computer Music Conference (ICMC)*, pages 193–197, IRCAM, France, 1984.
- [12] R. B. Dannenberg and N. Hu. Polyphonic audio matching for score following and intelligent audio editors. In *International Computer Music Conference*, pages 27–33, San Francisco, 2003.
- [13] I. Daubechies. *Ten lectures on wavelets*. SIAM, 1992.
- [14] Z. Duan, B. Pardo, and C. Zhang. Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions. *IEEE transactions on audio, speech, and language processing*, 18(8) :2121–2133, November 2010.
- [15] C. Duxbury, J.P. Bello, M. Davies, and M. Sandler. Complex domain onset detection for musical signals. In *6th Conference on Digital Audio Effects (DAFx-03)*, volume 1, pages 1–4, London, Angletaire, September 2003.

-
- [16] C. Duxbury, M. Sandler, and M. Davies. A hybrid approach to musical note onset detection. In *5th Conference on Digital Audio Effects (DAFx-02)*, volume 1, pages 33–37, Hamburg Germany, September 2002.
- [17] P. Fernandez-Cid and F.J. Casajus-Quiros. Multi-pitch estimation for polyphonic musical signals. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 6, pages 3565–3568, Seattle, Washington, USA, May 1998.
- [18] Harvey Fletcher. Auditory patterns. *Reviews of Modern Physics*, 12(1) :47–65, January 1940.
- [19] N.H. Fletcher. *Nonlinear dynamics and chaos in musical instruments*. IOS Press, January 1993.
- [20] D. Fober, S. Letz, and Y. Orlarey. Vemus - feedback and groupware technologies for music instrument learning. In *4th Sound and Music Computing Conference*, pages 117–123, Lefkada, Greece, July 2007. Soud music computing SMC.
- [21] B. Gagnon. Système d’alignement d’une partition de musique basé sur la déformation dynamique étendue du temps. Mémoire de maîtrise, Université de Sherbrooke, 2007. Sherbrooke, Québec, Canada.
- [22] B. Gagnon, R. Lefebvre, and C. Brunet. A high level musical score alignment technique based on fuzzy logic and DTW. In *Audio Engineering Society Convention 123*, New York, NY, USA, Oct 2007.
- [23] M. Gainza, B. Lawlor, and E. Coyle. Multi pitch estimation by using modified iir comb filters. In *ELMAR 47th International Symposium*, volume 1, pages 233–236, Zadar, Croatia, June 2005.
- [24] M. Good. MusicXML: an internet-friendly format for sheet music. In *XML conference & exposition*, pages 9–14, Orlando, Florida, December 2001.
- [25] M. Goto, H. Hashiguchi, T. Nishimura, and R. Oka. Rwc music database: popular, classical, and jazz music databases. In *In Proc. 3rd International Conference on Music Information Retrieval*, pages 287–288, 2002.
- [26] F. Harvey. Auditory patterns. *Reviews of Modern Physics*, 12(1) :47–65, January 1940.
- [27] D. J. Hermes. Measurement of pitch by subharmonic summation. *The Journal of the Acoustical Society of America*, 83(1) :257–264, January 1988.
- [28] N. Hu, R. B. Dannenberg, and G. Tzanetakis. Polyphonic audio matching and alignment for music retrieval. In *IEEE Workshop on Application of Signal Processing to Audio and Acoustics*, pages 185–188, New Paltz, New York, October 2003.
- [29] H. I-Hui and S. Kourosch. Imperfect pitch : Gabor’s uncertainty principle and the pitch of extremely brief sounds. *Psychonomic Bulletin & Review*, 23(1) :163–171, February 2016.
- [30] ISO. Iso 226 :2003. Organisation internationale de normalisation, 2003.
- [31] W.J. Jonathan Boo, Y. Wang, and A. Loscos. A violin transcriber for personalized learning. In *IEEE International Conference on Multimedia and Expo*, volume 2006, pages 2081–2084, Toronto, Ont., July 2006.
-

-
- [32] A. Kalpuri. Multipitch analysis of polyphonic music and speech signals using an auditory model. In *IEEE transactions on audio, speech, and language processing*, volume 16, pages 255–266, NJ, USA, February 2008. Institute of Electrical and Electronics Engineers, Piscataway.
- [33] E. J. Keogh and M. J. Pazzani. Derivative dynamic time warping. In *SIAM International Conference on Data Mining*, pages 1–11, Chicago, IL, 2001.
- [34] A. Klapuri. Sound onset detection by applying psychoacoustic knowledge. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 6, pages 3089–3092, Phoenix, AZ, USA, March 1999.
- [35] A. Klapuri. *Signal Processing Methods for the Automatic Transcription of Music*. PhD thesis, University of Technology, Tampere, Finland, 2004.
- [36] A. Klapuri. A perceptually motivated multiple-F0 estimation method. In *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, volume 1, pages 291–294, October 2005.
- [37] A. Klapuri. Multiple fundamental frequency estimation by summing harmonic amplitudes. In *7th International Conference on Music Information Retrieval*, International Society for Music Information Retrieval ISMIR, pages 216–221, Victoria, Canada, January 2006.
- [38] A. P. Klapuri. Multiple fundamental frequency estimation based on harmonicity and spectral smoothness. *IEEE Transactions on Audio, Speech, and Language Processing*, 11(6) :804–816, November 2003.
- [39] B. Li and Z. Duan. An approach to score following for piano performances with the sustained effect. *IEEE/ACM Transactions on Audio, Speech and language Processing (TASLP)*, 24(12) :2425–2438, December 2016.
- [40] A. Loscos, P. Cano, and Jordi Bonada. Low-delay singing voice alignment to text. In *International Computer Music Conference (ICMC)*, pages 437–440, Berlin, Germany, 1999.
- [41] A. Loscos, Y. Wang, and W. J. J. Boo. Low level descriptors for automatic violin transcription. In *7th International Conference on Music Information Retrieval*, International Society for Music Information Retrieval ISMIR, pages 8–12, Victoria, Canada, October 2006.
- [42] H. Lu, B. Zhang, Y. Wang, and W. K. Leow. iDVT: an interactive digital violin tutoring system based on audio-visual fusion. In *Proceeding of the 16th ACM international conference on Multimedia*, International Multimedia Conference, pages 1005–1006, New York, NY, USA, October 2008. Association for computing machinery (ACM).
- [43] A. Maezawa and H. G. Okuno. Bayesian audio-to-score alignment based on joint inference of timbre, volume, tempo, and note onset timings. *Computer Music Journal*, 39(1) :74–87, March 2015.
- [44] MakeMusic. *Finale 2012 Quick reference guide for windows*, 2012.
- [45] I. McLoughlin. *Applied Speech and Audio Processing : With Matlab Examples*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.
-

-
- [46] R. Meddis and M. J. Hewitt. Virtual pitch and phase sensitivity of a computer model of the auditory periphery. II: phase sensitivity. *The Journal of the Acoustical Society of America*, 89(6) :2883–2894, June 1991.
- [47] T. Miwa, Y. Tadokoro, and T. Saito. Musical pitch estimation and discrimination of musical instruments using comb filters for transcription. In *42nd Midwest Symposium on Circuits and Systems*, volume 1, pages 105–108, Las Cruces, NM, USA, August 1999.
- [48] S. Molla, I. Boulet, S. Meunier, G. Rabau, B. Gauduin, and P. Boussard. Calcul des indicateurs de sonie : revue des algorithmes et implémentation. In Société Française d’Acoustique SFA, editor, *10ème Congrès Français d’Acoustique*, pages –, Lyon, France, April 2010.
- [49] B.C.J. Moore. *An Introduction to the Psychology of Hearing*. Emerald, 2012.
- [50] M. Myerson. Tonara: the all-in-one musician’s companion, October 2013. <https://ipad.appstorm.net/reviews/music/tonara-the-all-in-one-musicians-companion/>, Accédé le 02 décembre 2019.
- [51] T. Nakamura, E. Nakamura, and S. Sagayama. Real-time audio-to-score alignment of music performances containing errors and arbitrary repeats and skips. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(2) :329–339, December 2015.
- [52] D. E. Newland. Harmonic wavelet analysis. In *Proceedings of the Royal Society of London, Series A*, volume 443, pages 203–225, London, October 1993.
- [53] A. M. Noll. Pitch determination of human speech by the harmonic product spectrum, the harmonic sum spectrum and a maximum likelihood estimate. In *Proceedings of the Symposium on Computer Processing in Communications*, volume 19, page 21, 1970.
- [54] N. Orio and F. Déchelle. Score following using spectral analysis and hidden Markov models. In A. Schloss, R. Dannenberg, and P. Driessen, editors, *International Computer Music Conference (ICMC)*, pages 151–154, Havana, Cuba, September 2001.
- [55] G. Peeters. Music pitch representation by periodicity measures based on combined temporal and spectral representations. In *ICASSP Acoustics, Speech and Signal Processing*, volume 5, pages V53–V56, 2006.
- [56] R. Plomp. *Aspects of tone sensation*, pages 1526–1532. Elsevier Academic Press, 1976.
- [57] E. Ponsot. *Global loudness processing of time-varying sounds*. Theses, Université Pierre et Marie Curie, December 2015.
- [58] J. G. Proakis and M. Salehi. *Communication systems engineering, second edition*, pages 629–634. Prentice-Hall, Inc., 2002.
- [59] S. Prokofiev. Concerto for violin and orchestra no. 1 op. 19 ; no. 2 op. 63 ; sonata for solo violin, op.115 ; violoniste : Gil shaham. Enregistrement audio, 1996.
- [60] M. Puckette. *The Theory and Technique of Electronic music*. World Scientific Publishing, 2007.
-

-
- [61] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, volume 77, pages 257–286, 1989.
- [62] I. P. Ramler and J. L. Chapman. Introducing statistical research to undergraduate mathematical statistics students using the guitar hero video game series. *Journal of Statistics Education*, 19(3) :1–21, 2011.
- [63] C. Raphael. Automatic segmentation of acoustic musical signals using hidden Markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(4) :360–370, 1999.
- [64] C. Raphael. A bayesian network for real-time musical accompaniment. In *Neural Information Processing Systems (NIPS)*, volume 14, pages 1433–1439, 2001.
- [65] C. Raphael. Orchestra in a box: a system for real time musical accompaniment. In *International Joint Conference on Artificial Intelligence IJCAI*, Acapulco, 2003.
- [66] S. Raptis. Imutus: interactive music tuition system. Répertoire en ligne des projets : Institute for Language and Speech Processing, 2005. <http://www.ilsp.gr/en/infoprojects/>, Accédé le 26 novembre 2019.
- [67] S. Raptis, A. Askenfelt, and D. Foer. IMUTUS - an effective practicing environment for music tuition. In *International computer music conference proceedings*, pages 377–387. International computer music association ICMC, 2005.
- [68] X. Rodet and F. Jaillet. Detection and modeling of fast attack transients. In *Proceedings of the International Computer Music Conference (ICMC)*, volume 1, pages 33–37, Cuba, September 2001.
- [69] F. J. Rodriguez-Serrano, J. J. Carabias-orti, P. Vera-Candeas, and D. Martinez-Munoz. Tempo driven audio-to-score alignment using spectral decomposition and online dynamic time warping. *ACM Transactions on Intelligent Systems and Technology (TIST) - Survey Paper, Special Issue: Intelligent Music Systems and Applications and Regular Papers archive*, 8(2) :22 :1–22 :20, 2017.
- [70] F. J. Rodriguez-Serrano, J. J. Carabias-orti, P. Vera-Candeas, N. Ruiz-Reyes, and F.J. Canadas-Quesada. An audio to score alignment framework using spectral factorization and dynamic time warping. In *16th International Conference on Music Information Retrieval*, International Society for Music Information Retrieval ISMIR, pages 26–30, Espagne, October 2015.
- [71] W. A. Schloss. On the automatic transcription of percussive music – from acoustic signal to high-level analysis. Technical report, Stanford University, 1985.
- [72] D. Schwarz, N. Orio, and N. Schnell. Robust polyphonic midi score following with Hidden Markov Models. In *International Computer Music Conference (ICMC)*, pages 1–4, Miami, USA, 2004.
- [73] N. Orio D. Schwarz. Alignment of monophonic and polyphonic music to a score. In *International Computer Music Conference (ICMC)*, pages 1–4, Havana, Cuba, 2001.
- [74] J.I. Shonle and K.E. Horan. The pitch of vibrato tonies. *Journal of the Acoustical Society of America*, 67(1) :246–252, 1980.
-

-
- [75] M. Slaney and R.F. Lyon. A perceptual pitch detector. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 357–360, Albuquerque, NM, USA, April 1990.
- [76] F. Soulez, X. Rodet, and D. Schwarz. Improving polyphonic and poly-instrumental music to score alignment. In *International Conference on Music Information Retrieval (ISMIR)*, pages 143–148, Washington, D.C., October 2003.
- [77] Y. Tadokoro and M. Yamaguchi. Pitch detection of duet using double comb filters. In *ECCTD'01 European Conference on circuit theory and design*, volume 3, pages 327–338, Espoo, Finland, August 2001.
- [78] C. Tait and W. Findlay. Wavelet analysis for onset detection. In *Proceedings of the 1996 International Computer Music Conference*, pages 500–503, San Francisco, 1996.
- [79] E. Terhardt. Pitch, consonance, and harmony. *The Journal of the Acoustical Society of America*, 55(5) :1061–1069, 1974.
- [80] G. Tzanetakis, G. Essl, and P. Cook. Audio analysis using the discrete wavelet transform. In *Proceedings of the Conference in Acoustics and Music Theory Applications. WSES*, pages 1–6, Greece, 2001.
- [81] A. Vivaldi. Vivaldi : Le quattro stagioni ; europa galante / fabio biondi ; l'arte di fabio biondi vol. 21 ; opus 111 ops 56-9120. Enregistrement audio, 1991.
- [82] Y. Wang, B. Zhang, and O. Schleusing. Educational violin transcription by fusing multimedia streams. In *Proceedings of the international workshop on Educational multimedia and multimedia education*, International Multimedia Conference, pages 57–66, Augsburg, Bavaria, Germany, September 2007. Association for computing machinery (ACM).
- [83] M. Yamaguchi and Y. Tadokoro. Pitch estimation of polyphonic songs using comb filters and singular value decomposition. In *TENCON Conference on Convergent Technologies for Asia-Pacific Region*, volume 1, pages 375–379, Bangalore, India, October 2003.
- [84] C. Yeh, A. Roebel, and X. Rodet. Multiple fundamental frequency estimation and polyphony inference of polyphonic music signals. *IEEE transactions on audio, speech, and language processing*, 18(6) :1116–1126, August 2010.
- [85] J. Yin, Y. Wang, and D. Hsu. Digital violin tutor: an integrated system for beginning violin learners. In *Proceedings of the 13th annual ACM international conference on Multimedia*, International Multimedia Conference, pages 976–985, Hilton, Singapore, November 2005. Association for computing machinery (ACM).
-