Sommaire

[Introduction 2](#_Toc444098379)

[1. Contexte et objectifs 2](#_Toc444098380)

[1.1. Contexte 2](#_Toc444098381)

[1.2. Objectifs et limites posées 4](#_Toc444098382)

[2. Méthode d’évaluation 5](#_Toc444098383)

[2.1. Critères d’évaluation 5](#_Toc444098384)

[2.2. Base de données annotée 8](#_Toc444098385)

[3. Méthodes d’analyse du signal 9](#_Toc444098386)

[3.1. Détection d’onset 9](#_Toc444098387)

[3.2. Estimation du tempo 21](#_Toc444098388)

[3.3. Analyse rythmique 31](#_Toc444098389)

[3.4. Analyse Harmonique 56](#_Toc444098390)

[4. Création d‘un logiciel à destination de l’utilisateur 62](#_Toc444098391)

[Étape 1 : Enregistrement 63](#_Toc444098392)

[Étape 2 : Choix du fichier audio 63](#_Toc444098393)

[Étape 3 : Paramétrage de la transcription 63](#_Toc444098394)

[Étape 4 : Choix du fichier de sortie et génération 64](#_Toc444098395)

[Utilisation de notre algorithme 64](#_Toc444098396)

[5. Améliorations possibles 65](#_Toc444098397)

[Conclusion 65](#_Toc444098398)

[Table des figures 66](#_Toc444098399)

[Références 67](#_Toc444098400)

# Introduction

# Contexte et objectifs

## Contexte

Depuis les débuts de l'informatique, la musique a su utiliser les nouveaux outils mis à sa disposition avec différents types de logiciel permettant d'assister le musicien, l'ordinateur constituant même de nos jours un instrument à part entière avec l'essor de la musique électronique et la MAO en général (Musique Assistée par Ordinateur).

En informatique musicale on distingue grossièrement deux types de logiciels. D'abords, les séquenceurs ou DAW (Digital Audio Workstation) qui permettent l'enregistrement, le montage, le séquencement et la gestion d'effets audio, parmi lesquels on peut citer Ableton, Cubase, Garage Band et FruitLoops. Ensuite, les logiciels d'édition de partition qui permettent au musicien de saisir de façon plus ou moins intuitive un morceau de musique et bien souvent de faire jouer cette partition par des banques de sons ou des sons synthétiques. Parmi ceux-ci, on peut citer Finale, Sibelius, LilyPond ou GuitarPro. Notons qu'une grande partie de ces logiciels est capable d'écrire une partition à partir d'un fichier MIDI importé.

### Problème posé

Ces derniers nécessitent systématiquement la saisie manuelle, par l'utilisateur - généralement un musicien - de toutes les notes de la partition. Cette saisie est souvent fastidieuse, d'autant plus quand la partition n'est pas triviale à écrire. Que ce soit dans le cas d'une composition (pour laquelle aucune partition n'existe encore) ou dans le cas d'une tentative de reproduction (pour laquelle aucune partition n'est accessible), de nombreux musicien souhaite créer une partition, soit de tête soit à partir d'un modèle audio qu'ils chercheront à reproduire, notamment grâce à la possibilité de jouer la partition.

Notre projet est donc de répondre à ce problème en facilitant le travail du musicien et en générant à partir d'un fichier audio, une partition aussi proche que possible de la réalité. Cette tâche est notamment connue sous le nom de « transcription automatique » d'un morceau de musique (*automatic music transcription*). L'enjeu est de détecter d'une part « quand » des notes sont jouées, que ce soit en déterminant le tempo, le début et la fin des notes. D'autre part, il faut détecter « quelles » notes ont été jouées (*pitch detection*).

### Schéma fonctionnel

Le schéma ci-dessous représente la composition de notre système de transcription automatique de la musique. On y repère différents blocs qui seront détaillés dans ce rapport. Ces blocs sont les suivants :

La détection d'onsets (*onset detection*) qui permet de déterminer le début d'une note. Quand deux notes se suivent consécutivement, le début de la seconde correspond à la fin de la première. Ce bloc inclus implicitement la détection d'offset (*offset detection*) c'est à dire la détection d'un silence entre deux notes jouées consécutivement.

La segmentation du signal audio vis à vis des onsets ET offsets pris dans l'ordre chronologique.

Le calcul du tempo à partir de la sortie de l'onset détection et faisant appel au bloc suivant :

L'analyse de la composition rythmique, permettant d'assigner à chaque segment, la durée musicale la plus probable compte tenu du tempo.

La construction de mesures régulière, en corrigeant ainsi d'éventuelles erreurs apparues dans le bloc précédent.

La détermination des tons, c'est à dire de la note dans la gamme tempérée.

La détermination des octaves, sans tenir compte du ton trouvé.

|  |
| --- |
| D:\rapport\images\Schéma fonctionnel générateur partition.png |
| Figure 1 - Schéma fonctionnel général de notre système de transcription automatique |

### État de l'art

Le problème est traité depuis les années 70 environ, notamment grâce à des transferts de solutions techniques venant du traitement de la parole. Cependant, en cherchant à résoudre le problème, on se heurte rapidement à différents problèmes [1].

Pour la musique polyphonique, ou la détection de plusieurs notes simultanées (*multi pitch detection*), aucune solution proposée à ce jour ne propose de résultats convaincants.

Concernant la détection de mélodie ou bien le cas de la musique monophonique, le timbre des instruments, la perception humaine des sons ou le bruit provoque encore des erreurs relativement inacceptables.

Concernant la détection de tempo, l'interprétation humaine est relativement subjective et pose donc problème.

Concernant la détection d'onsets, là encore l'oreille humaine est bien meilleure qu'un algorithme de nos jours et est capable de percevoir des nuances très subtiles dans l'apparition des notes que les algorithmes manqueront.

Globalement, de nombreuses méthodes existent pour tenter de résoudre tous ces problèmes. Notre travail a majoritairement consisté à chercher les méthodes pour lesquelles nous obtenions les meilleurs résultats, tentant d'améliorer une solution existante (dans le cas de l'estimation de tempo), voire en créant complètement notre méthode selon notre expérience et notre intuition (cas de l'analyse et la correction rythmique).

Les résultats que l'on peut espérer seront loin d'être parfait d'autant que les erreurs entre les algorithmes se combineront entre elles. En effet, une note ne sera considérée comme bien détectée que si on trouve le bon ton, à la bonne octave et correspondant à la bonne durée jouée à un tempo bien estimé !

### Solutions analogues

Certains logiciels remplissant plus ou moins cette tâche sont diffusés à titre commercial ou gratuit et disponibles depuis les années 2000 environ. Parmi eux, on peut citer les logiciels/applications reconnaissant les accords : Chordec, AnySong Chord Recognition, Mémo musical (de Apple).

Ces logiciels permettent de transcrire ce que l'on peut décrire comme une partition simplifiée incluant la notion de tempo, de rythmique mais d'un point de vue harmonique, on se contente de détecter un accord c'est à dire une classe de notes jouées simultanément sans notion d'octave. Ces logiciels sont conçus pour la musique monophonique mais contenant plusieurs notes jouées en même temps (un instrument, plusieurs notes à la fois).

D'autres logiciels traitent plus directement notre problème. On peut citer de nombreuses solutions labellisées comme des « convertisseurs mp3 vers MIDI », ce qui correspond à ce que nous cherchons à faire, le format MIDI permettant bien de pouvoir obtenir une partition par la suite. On peut également citer les logiciels payants Intelliscore et Sibelius qui adressent directement notre problème au sein d'un éditeur de partition.

## Objectifs et limites posées

L'objectif de ce projet de fin d'étude est de proposer un système de transcription automatique de bonne qualité, c'est à dire qu'il faut que la partition créée permette de reconnaître le morceau dont elle est issue.

On l'a dit, la transcription musicale automatique ne peut pas être réalisée au sens strict du terme dans l'état de l'art actuel. Pour s'en approcher, il faut réduire son cadre avec certaines limitations et optimiser le système pour obtenir les meilleurs résultats possibles dans ce cadre. Pour cela nous nous posons, dans ce projet les contrainte suivantes :

L'algorithme de *pitch-detection*, que nous appellerons Analyse Harmonique, sera **optimisé pour la guitare**. Cela signifie que seules les notes allant du E2 (82.4Hz) au D #6 (1244.5Hz) pourront être détectées. Le choix de la guitare comme instrument exclusif aura également un impact sur la détection d'onset.

Cet algorithme ne traitera également que le problème de la détermination d'**UNE SEULE** note jouée à la fois.

L'estimation de tempo se fera dans l'intervalle de **55 BPM et 180 BPM.**

La détermination des durées musicales, que nous appellerons Analyse Rythmique, se fera pour un tempo donné. La durée musicale **la plus courte possible sera la double croche**, toute les autres durées musicales possibles seront des multiples de la durée d'une double-croche (croche, croche pointée, noire, etc.)

L'analyse rythmique sera également optimisée pour construire des **mesures 4/4** (soit l'équivalent de 4 noires par mesure)

La détection d'offsets (de fin de notes, c'est à dire de l'apparition de silences) ne sera pas évaluée et ne fera pas l'objet d’exigences de qualité.

# Méthode d’évaluation

## Critères d’évaluation

Pour mesurer les performances des différents algorithmes que nous avons développés, nous avons mis en place différents indicateurs spécifique à chaque partie du projet. Ces indicateurs nous ont permis de comparer deux algorithmes ou versions différentes et ainsi sélectionner les meilleures solutions.

### Onset Detection :

Concernant l’Onset Detection, nous avons comparé le nombre et les instants des onset détectés comparés avec les onsets attendus. La stratégie est d’appairer les onsets détectés et annotés qui sont proches deux à deux. S’il reste des onsets non appairés, il s’agit de fausses détections ou de détections manquées selon que l’onset seul est respectivement détecté ou annotés. Les offsets ne sont ni annotés ni évalués.

|  |
| --- |
| annotation_onsets |
| Figure 2 - Exemple de confrontation entre les onsets annotés et les onsets détectés |

Au final on récupère 3 indicateurs, le taux de fausses détection, le taux de détections manquées et le taux de réussite :

|  |  |
| --- | --- |
| Taux de fausses détections : |  |
| Taux de détections manquées : |  |
| Taux de réussite: |  |
| Où # dénote le « nombre de … » |  |

### Analyse Rythmique :

Concernant l’Analyse Rythmique, on se base sur le même algorithme que pour l’onset détection au niveau de l’appairage. Pour chaque couple annotation-détection, on compare les durées de la note correspondante. Si cette note est suivie d’un silence car un offset a été détecté directement après, on ajoute la durée du silence à la durée de la note détecté :



Ainsi il n’y a aucun silence à annoter. Les notes issues d’une fausse détection d’onset ne sont pas évaluées, tout comme les détections d’onsets manquées ne sont pas prise en compte.

On calcule ensuite la matrice de confusion et utilise le taux de comparaison exacte comme indicateur de performance de l’AR. Cet indicateur est donc relatif au taux de réussite pour l’OD.

|  |
| --- |
| confusion_durees |
| Figure 3 - Exemple d'une matrice de confusion de la détermination des durées (AR) |

### Analyse Harmonique :

Concernant l’analyse harmonique, on utilise exactement le même procédé que pour l’analyse rythmique cependant, il n’y a pas de fusion avec les notes « silencieuses », elles sont simplement ignorées. On compare séparément les octaves et les tons (do, do#, ré, etc…).

Là encore on garde pour indicateur, le taux de comparaison exacte de la matrice de confusion. Cet indicateur est lui aussi relatif au taux de succès d’onset detection.

|  |
| --- |
|  |
| Figure 4 - Exemple de matrices de confusions de la détermination des notes (AH) |

## Base de données annotée

Pour évaluer les performances de nos algorithmes selon les indicateurs décrits précédemment, nous avons besoin de constituer une base de données annotés et répondant à nos critères.

Ajout d’un morceau à la base :

Enregistrement d’un son (morceau joué à la guitare) dans des conditions optimales (micro interne du PC relié directement à l’amplificateur). L’enregistrement doit être effectué en rythme (au métronome), sans erreurs ou approximation. Ces conditions sont aussi valable dans le cas où un tiers utiliserait l’application.

Rédaction manuelle d’une partition (au format GP4 ou 5) servant de référence. Cette partition correspond au morceau effectivement joué et non au morceau original qui a été reproduit quand c’est le cas.

Ajout du morceau aux scripts et lancement de l’algorithme actuel d’onset detection.

La sortie de cet algorithme est une liste d’onset erronées. Il faut ensuite manuellement ajouter les onsets manquants et retirer les onsets faussement détectés en se basant sur la partition.

Rédaction du fichier « expected.txt » qui sera lu par l’algorithme d’évaluation. Ce fichier est construit comme suit (un élément décrit [] correspond à 1 caractère ou un entier. Les références de solfège sont en notation anglo-saxonne).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Ligne | Fichier « expected.txt » | | | | | | |
| 1 | [#notes] | | | | | | |
| 2 | [tempo] | | | | | | |
| 3 | [Échantillon de l’onset n°1] | \t | [durée de la note en nombre de double croche] | \t | [Ton (A…G)] | [# ou espace] | [octave de la note] |
| 4 | Idem pour la note n°2 | | | | | | |
| Etc… | | | | | | | |

Toutes ces manipulations prennent du temps à effectuer (environ une demi-journée par morceau).

C’est la raison pour laquelle notre base de données est si restreinte. D’autre part, avant la mise en place de cette méthode et notamment l’étape n°1, nous avions inséré dans notre base un morceau issu directement de la version CD (les 8 premières secondes de Day Tripper où uniquement de la guitare est jouée.) Cependant ce morceau est trop court pour qu’une bonne estimation du tempo soit faite. Nous avions également inséré des morceaux générés à partir de la banque de sons du logiciel Guitar Pro. Ces sons sont synthétiques et les partitions sont très simples. Notre logiciel obtient donc des résultats presque parfait (aux différences d’interprétation près comme le tempo). Cela prouve qu’en théorie les solutions choisies sont les bonnes et que les erreurs sont introduit par l’instrument réel et le musicien réel (différences d’attaques, de timbres, d’enveloppe, erreurs de rythme, etc…). Notre logiciel est donc capable d’effectuer l’opération inverse de Guitar Pro à savoir : « créer une partition à partir d’un son synthétique ». Ces bons résultats ont été obtenus tôt dans notre projet, et notre travail par la suite a été d’améliorer les performances sur des morceaux réels.

# Méthodes d’analyse du signal

## Détection d’onset

Cette partie traite de la détection d’onset. Elle correspond au tout premier bloc du schéma fonctionnel.

Un onset fait référence au début d’une note de musique ou d’un autre son, du moment que l’amplitude du signal passe de zéro à un pic initial. Ainsi, après avoir isolé ces notes unes à unes, nous pourrons faire une analyse des tons et des octaves pour chaque note.

|  |
| --- |
| /Users/apple/Desktop/Capture d’écran 2016-01-29 à 10.43.03.png |
| Figure 5 - Onset |

Il existe une dizaine de méthodes de détection d’onsets. Cependant, nous ne les avons pas toutes codées. Celles-ci sont directement appliquées au signal audio. Il en résulte un signal possédant plusieurs pics de plus ou moins fortes amplitudes. Enfin, une fonction de localisation des pics est nécessaire pour extraire ces fameux onsets. Au cours de ces étapes, il est possible qu’il se produise des fausses détections ou des onsets non-détectés.

Voici le schéma général d’un algorithme de détection d’onset :

|  |
| --- |
| /Users/apple/Desktop/Capture d’écran 2016-01-29 à 10.43.30.png |
| Figure 6 - Algorithme de détection d'onset |

Dans notre projet, nous avons testé plusieurs fonctions de détection d’onset puis nous en avons combiné pour améliorer leur détection. Voici les principales approches et des explications correspondant aux méthodes que nous avons implémentées.

### Méthodes liée aux caractéristiques temporelles du signal audio

#### Energie du signal

La méthode la plus évidente et simple prendrait en compte les variations d’énergie du signal audio (sans transformation) pour détecter les onsets. Cette méthode est efficace pour des signaux comme des percussions, qui présentent une forte amplitude dès lors qu’un son est produit.

On pourra par exemple prendre la norme au carré d’une trame de signal, ce qui nous donnerai son énergie, faire la dérivée première pour voir les variations d’énergie sur le signal. Enfin prendre un seuil à partir duquel l’énergie est suffisamment grande pour pouvoir déterminer un onset.

**Inconvénient :**  Ce type de méthode ne fonctionne pas de façon optimale car certains onset ne sont pas détectés par rapport au seuil que l’on se fixe (fausse détection/sur-détection). De plus, dans le cas de notre projet qui prend en compte les cordes de la guitare, les onsets produits par celle-ci n’ont pas une variation en amplitude comparable à des instruments percussifs.

### Méthodes liées aux caractéristiques spectrales du signal audio

#### Spectral Flux ou Spectral difference (domaine fréquentielle)

Cette méthode se rapproche de celle présentée précédemment, mais elle prend cette fois-ci en compte un outil : la transformée de Fourier discrète.

On commence par faire la transformée de Fourier du signal audio. Dans notre projet, nous avons découpé le signal audio en plusieurs séquences. Ce sont ces séquences qui seront analysées et traitées les unes à la suite des autres.

La transformée de Fourier nous donne pour un signal audio en entrée une bande de fréquence en ordonnée et le temps, représenté par des échantillons k, en abscisse.

Fréquences

Matrice de nombre complexes

Echantillons k du signal

Voici un spectrogramme, représentant de façon intuitive la présence des fréquences dans le temps d’un morceau de 25 secondes :

|  |
| --- |
|  |
| Figure 7 - Spectrogramme d'un morceau de 25 secondes |

La matrice contient des nombres complexes, porteurs de l’information de la phase et de l’amplitude pour chaque échantillon k et pour chaque fréquence. Cette transformée de Fourier peut être définie comme un groupe d’oscillateurs sinusoïdaux de différentes fréquences avec des amplitudes et phases variant dans le temps.

Cette méthode détermine la variation de l’amplitude en puissance du signal pour chaque échantillon de fréquence (on ne prend plus l’énergie du signal audio mais l’amplitude de la transformée de Fourier).

La valeur absolue de la transformée de Fourier (amplitude) est dérivée pour obtenir la variation pour chaque fréquence entre l’échantillon k et k-1. On somme ensuite pour chaque échantillon de temps k ces variations pour pouvoir avoir une représentation temporelle de ces changements.

Une forte variation en amplitude à instant k nous donnera donc un pic.

**Inconvénient :** L’efficacité de cette méthode décroit pour des signaux non-percussifs (ce qui est également le cas pour la méthode présentée précédemment).

#### Phase deviation (domaine de la phase)

Contrairement à la méthode du spectral flux qui prend en compte la puissance du signal, la méthode de déviation de phase utilise la variation dans la phase du signal.

Elle fonctionne de la même manière que spectral flux sauf que l’on prend la phase du signal (angle) suite à la transformée de Fourier (que l’on recadre entre –π et π). On étudie donc la phase du signal pour une fréquence donnée sur l’ensemble des échantillons k du signal.

La première différence de phase entre un échantillon k et k-1 nous donne la fréquence instantanée (première dérivée). Le changement de variation de la fréquence instantanée (dérivée seconde) est un indicateur d’un possible onset. En effet, au cours de la partie à l'état stable du signal, ces oscillateurs (voir FFT) ont tendance à avoir des fréquences constantes. Par conséquent, la différence entre deux valeurs de phase consécutives doit rester constante.

La variation de cette différence de phase nous donnera l’indication d’un possible onset (voir (1) dans la bibliographie).

#### Complex domain

L’idée de cette méthode est de considérer simultanément les effets de ces deux grandeurs, l’amplitude et la phase, en prédisant des valeurs dans le domaine complexe. On calcule l’amplitude et la phase d’un échantillon (valeur cible) à partir de deux échantillons précédents en supposant que l’état est stable.

La valeur cible est définie par la forme polaire du nombre complexe correspondant à un échantillon de la FFT comme :

Capture%20d’écran%202016-02-01%20à%2009.48.06.png

= (amplitude de l’échantillon k-1) \*exp(somme entre la phase de k-1 et la différence de phase entre k-1 et k-2).

Capture%20d’écran%202016-02-01%20à%2010.05.18.png

Cette valeur est donc estimée avec une amplitude et un taux de changement de phase constant.

La valeur courante possède l’amplitude et la phase correspondant à son échantillon.

**

En mesurant la distance euclidienne entre la valeur cible et la valeur courante dans le domaine complexe, on est capable de mesurer la stationnarité pour le kème échantillon.

La déviation de phase est donnée par la formule suivante (n représente k) :



L’égalité de droite correspond à la variation de la fréquence instantanée de l’échantillon k et celle de gauche correspond à la variation de la fréquence instantanée de l’échantillon k- 1.

La déviation de phase courante est alors donnée par :



La phase de la valeur ciblée est pour rappel :

Capture%20d’écran%202016-02-01%20à%2010.05.18.png

En simplifiant cette distance euclidienne (notamment en forçant la formule de la cible visée sur l’axe des réels, voir doc 4 sur l’onset detection), on arrive à la formule suivante :

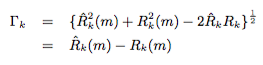


|  |
| --- |
| Capture%20d’écran%202016-02-01%20à%2010.40.26.png |
| Figure 8 - Simplification de la distance euclidiènne |

Lorsque la différence de phase entre deux échantillons successifs est nulle, c’est à dire quand l’égalité suivante est respectée :



alors :

****

Par conséquent, lorsque la valeur de la différence de phase est nulle (la prédiction est bonne), la différence complexe se réduit à la différence d’amplitude (différence d’énergie). Quand la différence de phase est non-nulle, un terme additionnel s’ajoute qui prend en compte la déviation de phase.

Finalement, on somme le terme Γk pour tous les échantillons k.

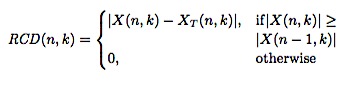
Capture%20d’écran%202016-02-01%20à%2011.27.58.png

Cette méthode donne une détection plus robuste que celles testées précédemment ( [2], [3], [4], [5]).

#### Rectified Complex Domain

Le problème avec la méthode Complex Domain est qu’elle ne fait pas de différence entre les augmentations et les diminutions de l’amplitude du signal, donc les onsets ne sont pas différenciables des offsets.

La méthode Rectified Complex Domain prend en compte les cas où la différence complexe représente une augmentation d’énergie, et donc un onset.

****

(Voir [3], [5]).

### Autres types de méthodes

#### Méthodes liées à des analyses temps-fréquence/échelle de temps différentes (alternatives à la FFT).

Il existe d’autres alternatives aux méthodes d’analyse des coefficients de Fourier ou de l’énergie du signal comme l’utilisation d’échelles temps-fréquences différentes.

Une méthode utilise par exemple la décomposition en ondelettes (ondelette de Haar). Une autre est basée sur une représentation temps-fréquence de Cohen.

Cependant, nous n’avons pas développé ces méthodes car aux vues des résultats présentés dans les études réalisées sur l’onset detection, les méthodes mentionnées ci-dessus donnent déjà de très bons résultats. [2], [6], [7].

#### Probability Models

Une autre alternative possible que nous n’avons pas utilisée est de recourir aux méthodes statistiques. Elles sont basées sur l'hypothèse que le signal peut être décrit par un modèle de probabilité.

Un système peut alors être construit qui fait des analogies de probabilités sur les temps supposés des changements brusques dans le signal, étant donné les observations disponibles.

Le succès de cette approche dépend de la proximité de l'ajustement entre le modèle supposé, donc la distribution de probabilité décrite par le modèle, et la distribution des données observée, et peuvent être quantifiés utilisant des mesures de probabilité.

Ces modèles peuvent par la suite être perfectionnés par des techniques d'apprentissage automatique comme les réseaux de neurones. [8]

### Algorithme de détection des pics

Comme expliqué précédemment, on travaille échantillon par échantillon sur le signal. On commence par faire une transformée de Fourier discrète sur l’échantillon, on applique la fonction d’onset detection, puis on applique un filtre coupe-bande pour éliminer certains parasites sur le résultat de la fonction d’onset.

Après plusieurs tests avec les méthodes présentées ci-dessus, nous avons décidé de combiner deux méthodes. En effet on a pu constater que la méthode « Spectral flux » a tendance à oublier certaines onset alors que la méthode « Complex domain » au contraire nous donne des faux-positifs. Nous avons donc affecté à chacune de ces deux méthodes (que nous avons normalisé) un coefficient de pondération pour obtenir une fonction d’onset detection plus efficace.

Une fois la courbe représentant la fonction d’onset detection obtenue, nous avons établi une fonction donnant la moyenne locale sur le signal pour pouvoir récupérer les pics qui se trouvent au-dessus de cette moyenne.

### Comparaisons de différentes méthodes

Ci-dessous la comparaison entre deux fonctions d’onset detection sur une même chanson (avec un léger décalage dans le temps). On note que la méthode Complex domain donne des pics plus précis.

|  |
| --- |
| ../../test/CD_specFlux.jpg |
| Figure 9 - Comparaison de deux méthodes (Spectral flux & Complex domain) |

Ci-dessous la comparaison entre deux fonctions d’onset detection sur une même chanson (avec un léger décalage dans le temps). On constate que la méthode Complex domain donne des pics là encore plus précis, la méthode de phase deviation donnant des pics de faibles amplitudes et bruités.

|  |
| --- |
| ../../test/cd_phase_d.jpg |
| Figure 10 - Comparaison de deux méthodes (Phase deviation & Complex domain) |

Comme expliqué dans la partie « Peak detection algorithm », nous avons décidé de combiner les deux méthodes SpectralFlux et Complexe domain. Le résultat est le suivant :

`

|  |
| --- |
| ../../test/combinaison_methodes.jpg |
| Figure 11 - Combinaison de méthodes (Spectral flux & Complex domain) |

### Détection d’offset

La détection d’offsets est le pendant à la détection d’onset. Il s’agit de détecté la fin d’une note quand celle-ci est suivi d’un silence. En effet quand deux notes s’enchaînent, il est inutile d’essayer de déterminer la fin de la première car on considèrera qu’il s’agit du début de la seconde (à un échantillon près). Quand une note n’est pas suivie cependant, il n’y a pas d’onsets détecté à la fin de celle-ci ce qui nous empêche de déterminer la durée de la note. C’est la raison pour laquelle nous avons besoin de détecter un offset (une fin de note) uniquement dans le cas où cette note est suivie d’un silence.

La méthode que nous utilisons est assez intuitive et nous l’avons conçu nous-même. Elle consiste en la recherche d’une chute « drastique » du signal d’onset. Cependant cette notion est trop présente dans le signal d’onset pur, nous utilisons donc sa moyenne locale, déjà calculée pour la détection de pics, qui constitue une version filtrée (passe-bas).

Pour détecter une chute « drastique », nous cherchons simplement des minima locaux dans la dérivée de la moyenne de la fonction d’onset. Nous ne conservons que les minima dont la valeur absolue est supérieure à l’écart-type de cette dérivée.

|  |
| --- |
|  |
| Figure 12 - Exemple de détection d'onset sur "Hardest Button to Button" |

Les offsets détectés sont assez mal localisé dans le temps. De plus nous ne les avons pas annotés car la notion de fin de note est beaucoup plus subjective pour un auditeur que son apparition. Nous nous en servons surtout pour empêcher certaines durées de notes d’être trop longue. Lors de l’évaluation du rythme, on ne les prend pas en compte et l’on considère les durées d’onset à onset.

De plus, considérant l’analyse harmonique, les notes commençant par un offset seront automatiquement classée comme des silences (*Rest* en anglais) et donc l’algorithme ne testera pas cette partie du signal audio.

### Résultats

Notre fonction d’onset detection se présente dans sa forme finale comme ceci :

|  |
| --- |
| ../../test/final.jpg |
| Figure 13 - Algorithme final d'onset detection |

La courbe jaune est une moyenne locale, les pics rouges sont les onsets, les violets des offsets.

|  |
| --- |
|  |
| Figure 14 - Evolution des résultats de la détection d'onset |

|  |
| --- |
| ../../Capture%20d’écran%202016-02-22%20à%2010.29.10.png |
| Figure 15 - Comparaison des pourcentages de détection d'onset par méthodes |

Après avoir essayé de nombreuses méthodes, la combinaison du SpectralFlux et de Complex domain nous semble la plus appropriée pour notre application. Nous arrivons à un taux de détection d’onset de l’orde de 97%.

**Principaux problèmes rencontrés :**

* Sur-détections
* Non-détections
* Lissage des fonctions d’onset
* Moyenne locale
* Onset trop rapprochés dans le temps

## Estimation du tempo

Puisque nous souhaitons transcrire les événements musicaux du domaine temporel réel vers un domaine rythmique « virtuel », nous devons pouvoir effectuer une conversion entre une note (intervalle entre deux onsets détectés) dont la durée se mesure en seconde et une note dont la durée est relative aux autres notes du morceau. Cette conversion s'effectue via la notion de tempo.

Qu'est-ce que le tempo ?

Le *tempo*, aussi appelé « unité de temps » représente la durée d'un temps musical élémentaire (noire le plus souvent), c'est à dire la vitesse d'exécution d'un morceau ou sa cadence. Dans la musique moderne, il est exprimé en BPM (battement par minute). Un tempo de 120 BPM signifie que le musicien joue l'équivalent de 120 noires en une minute.

Pour pouvoir transcrire notre morceau, nous avons donc besoin de déterminer une valeur pour ce tempo. Dans certains cas, l'utilisateur sait déjà à quel tempo il joue, soit s'il respecte déjà une partition existante, soit s'il joue au métronome. Notre logiciel propose donc la possibilité pour l'utilisateur de spécifier cette valeur.

Dans bien d'autre cas, l'utilisateur ne sait pas exactement à quel tempo il a joué le morceau enregistré, soit parce qu'il s'agit d'une improvisation, soit parce qu'il joue un morceau de tête. Nous avons donc besoin dans ce cas d'estimer une valeur du tempo qui corresponde au mieux à l'enregistrement.

L'estimation (ou détection) de tempo est un problème déjà largement traité dans le domaine de l'extraction d'information musical (*musical information retrieval*) et fait même l'objet d'un concours Musical Information Retrieval Evaluation Exchange (MIREX) depuis sa première édition en 2005.

Cependant, la détermination d'un tempo est une tâche difficile même pour un humain voire un musicien. Par exemple, dans le cas classique d'une marche militaire jouée = 120 BPM (mesure 4/4 classique, dont les temps sont très marqués), certaines personnes estimeront que le tempo correspond à 60BPM en ne prenant en compte d'un temps sur deux. De même pour une valse en 3 temps jouée à 150 BPM, on trouvera certaines estimation) 50BPM en ne comptant qu'un temps sur 3 [9]. L'effet est d'autant plus fréquent que les temps du morceau sont peu marqués.

* + Nous proposons ici une méthode prenant en compte nos restrictions et nos avantages.
  + Nous nous bornons à prendre en compte des mesures 4/4, dont le tempo est considéré dans l'intervalle 55 BPM à 180 BPM.
  + Nous ne considérons pas les durées musicales inférieures à la double-croches. Nous disposons déjà d'une méthode d'onset détection.
  + Nous sommes capable, via l'algorithme de normalisation des durées de notes d'estimer pour chaque durées réelles, la certitude avec laquelle on peut la convertir dans une durée musicale donnée, sachant un certain tempo (Cf. la fin de la description de la méthode).

### Estimation de tempo par fenêtrage, autocorrélation et corrélation avec des trains d’impulsions.

La méthode de détermination du tempo que nous avons mis en place est en grande partie issue de la publication *« ﻿Streamlined Tempo Estimation Based on Autocorrelation and Cross-correlation With Pulses »* [9], que nous appellerons à présent la publication. Cette méthode, comme d’autres méthodes de détermination du tempo, est basé sur le principe que le tempo est une périodicité de l’apparition des notes. Le signal d’onset qui correspond à cette notion d’apparition des notes est donc pseudo périodique et le tempo du morceau de musique peut être déduit de cette période.

|  |
| --- |
|  |
| Figure 16 - Puissance d'onset des 10 premières secondes de "No Surprises" |

#### Fenêtrage

La première étape consiste à fenêtrer le signal d’onset OSS en segment d’environ 5.5 secondes de long et se chevauchant de 10% (soit 0.55 secondes en commun entre deux fenêtres consécutives).

Ce fenêtrage a pour but de rendre indépendante les différentes parties du morceau. Imaginons qu’au bout de 20 secondes, le morceau contienne un long silence de plusieurs secondes. La périodicité que l’on cherche à déterminer serait complétement rompu. En fenêtrant comme décrit plus haut, on peut tenter de déterminer le tempo sur les 30 premières fenêtres obtenues (5.5 secondes + 29x0.55 seconde ≈ 20 secondes). On final on obtient M fenêtre dénotées *m*.

#### Autocorrélation généralisée

Pour chaque fenêtre obtenue, on calcul ensuite l’autocorrélation généralisée. Ce calcul se fait en passant par le domaine fréquentiel (par FFT), en compressant le spectre obtenu par un coefficient *c*, puis à revenir dans le domaine temporelle (IFFT) :

Avec c=2, on retrouve la formule de l’autocorrélation classique qui est la transformée de Fourier de la DSP du signal temporelle. Pour un *c* plus faible, les pics de l’autocorrélation sont plus étroit (plus précis). Comme dans la publication, nous utilisons un *c=0.5*.

|  |
| --- |
|  |
| Figure 17 - Autocorrélation généralisée sur la première fenêtre de "No Surprises" |

Renforcement des harmoniques :

Les pics que l’on observe dans l’autocorrélation représentent la périodicité du signal d’onset. On remarque cependant qu’ils sont espacés de façon régulière les uns des autres. On cherche donc à intensifier ce phénomène en multipliant l’autocorrélation obtenue avec une version d’elle-même sous-échantillonnée (ou accélérée) d’un rapport de 2, puis de 4 :

|  |
| --- |
|  |
| Figure 18 - Autocorrélation aux harmoniques augmentées de la 1ère fenêtre de "No Surprises" |

#### Sélection de pics

Dans le signal EAC décrit précédemment, on sélectionne ensuite les 10 plus forts pics dans l’intervalle [0.33 s, 1.09s] correspondant aux bornes que l’on se fixe pour détecter un tempo (55 BPM et 180 BPM). On déduit du retard de ces pics une liste de 10 tempos candidats (via le retard associé P).

#### Corrélation du signal d’onset avec des trains d’impulsions

Dans le but de déterminer quel candidat est le tempo présent dans la fenêtre, on construit pour chaque candidat un train d’impulsions construit comme suit :

On place dans un train d’impulsions, 4 impulsions aux instants de retard correspondants à BxP secondes, pour B allant de 0 à 3 et P étant le retard déduit de l'étape précédente.

On construit 2 autres trains d’impulsions de la même façon mais avec 1.5P et 2P. On somme tous ces trains d’impulsions avec un rapport de deux en faveur du premier train. On récupère ainsi 10 trains d'impulsions correspondant chacun à un retard candidat P.

|  |
| --- |
|  |
| Figure 19 - Construction des trains d'impulsions |

On calcule ensuite la corrélation entre un train d'impulsions et le segment de la fonction d'onset correspondant. Cette corrélation permet d'apprécier, pour un retard donné et un tempo donné, la valeur de la fonction d'onset « dans une noire », « dans une croche », etc.

On récupère un signal dépendant du retard (puisque c'est une corrélation) et spécifique à un tempo candidat. Pour chacun des signaux résultants, on calcule la variance et le maximum que l'on somme et normalise selon les retards P de façon à obtenir un score pour chaque retard P. On décide alors que le tempo qui correspond à cette fenêtre *m* est celui dont le retard a le plus fort score que l'on note *pm*.

On trace ensuite une gaussienne en fonction du tempo en BPM, centrée en *pm* dont l'écart-type correspond à 10BPM. Le choix d'une gaussienne permet d'apprécié de légère variations dans l'évaluation d'un tempo

#### Détermination du meilleur candidat.

On exécute toutes les étapes précédentes pour toutes les fenêtres *m*, résultant en M gaussienne représenté sur la figure suivante. À ce stade, on pourrait imaginer la prise en compte de variations importantes du tempo comme une accélération crescendo. Nous n'avons pas implémenté cette possibilité mais il s'agit d'une perspective d'amélioration. Nous nous contentons de chercher à estimer le tempo représentant l'intégralité du morceau.

|  |
| --- |
|  |
| Figure 20 - Probabilité qu'un tempo soit représentatif d'une fenêtre |

On moyenne ensuite ces gaussiennes selon le tempo pour obtenir un vecteur C correspondant à la densité de probabilité qu'un tempo soit représentatif du morceau complet.

|  |
| --- |
|  |
| Figure 21 - Probabilité qu'un tempo soit représentatif du morceau |

#### Méthode originale décrite dans la publication

Détection des 3 pics principaux (même s’il en existe moins…). Le pic ayant la plus forte amplitude correspond au tempo candidat L. Les deux autres pics sont le deuxième et troisième meilleur candidats, L2 et L3.

Utilisation d’une Support Vector Machine (SVM) :

En entrée de la SVM, il est demandé plusieurs paramètres. Le tempo sera détecté avec une certaine incertitude σ. Dans la publication, cette incertitude varie avec le tempo candidat L : . Dans notre implémentation elle est fixe à 4 BPM. Les paramètres sont les suivants (tempo = tempo réel)

La probabilité que le tempo soit inférieur à L :

La probabilité que le tempo soit supérieur à L

La probabilité que le tempo soit proche de L

La probabilité que le tempo soir proche de L/2 (rapport de ½)

La probabilité que le tempo soir proche de 2L (rapport de 2)

La somme de ces 3 dernières probabilités

Les rapports entre L et le deuxième meilleur candidat

Le rapport entre L et le 3ème meilleur candidat

Le nombre de tempo pour lequel la densité de probabilité n’est pas nulle (en mode discret)

Le tempo candidat lui-même, L.

Ces paramètres sont ensuite normalisés entre 0 et 1 et on applique à ces 10 paramètres, les poids de la SVM déterminés au préalable par apprentissage. Si le produit scalaire entre le vecteur 10x1 des paramètres et le vecteur 10x1 des poids de la SVM est positif, il faut doubler le tempo candidat.

Remarque : Cette méthode bien qu’elle donne des résultats convenables, nous semble assez étrange dans ses détails : la publication ne cite que les paramètres 1, 4 et 10 alors que dans l’implémentation Matlab donnée en référence apparaissent ces 10 paramètres différents.

D’autre part, nous avons pu récupérer les valeurs déterminées issues de l’apprentissage mais ces valeurs donnent de mauvais résultats sur notre jeu de donnée. Il a fallu les modifier manuellement et neutralisé certains paramètres pour obtenir des résultats corrects (méthode impartiale).

Enfin, la SVM de cette algorithme ne permet de déterminé que s’il faut doubler ou non le candidat final et non s’il faut le diviser par deux ou tout simplement choisir un autre candidat.

#### Amélioration proposée

##### Probabilité des durées de notes

Ce qui motive notre volonté de modifier l’algorithme est notamment le fait qu’il ne peut choisir qu’entre 2 tempo possibles, le meilleur candidat et son double. Dans bien des cas, le tempo réel correspondra à la moitié du meilleur candidat, ou bien à un double/moitié d’un autre candidat :

|  |
| --- |
|  |
| Figure 22 - Exemple de mise en échec de la méthode proposée |

La différence entre notre système et celui de la publication est que nous nous servons par la suite du tempo pour déterminer la durée musicale des notes (croche, noire, blanche…). Nous avons donc besoin d’un tempo fiable pour réaliser correctement cette étape. Une erreur sur le tempo à ce moment entraîne des répercussions par la suite. Dans notre exemple, une noire jouée à 90BPM sera déterminée comme une croche pointée à 120 BPM, ce qui est déjà en soi une erreur. Cette erreur aura également un impact au moment du découpage en mesure 4/4 prévu par notre algorithme.

Or cette étape de détermination des durées est également un atout que nous pouvons utiliser. En effet, l’objectif de l’algorithme de détermination des durées est d’assigner à chaque note jouée une durée musicale ainsi qu’un coefficient de certitude que la durée a bien été déterminée. Ce coefficient correspond à la probabilité que la note jouée soit d’une certaine durée musicale, sachant le tempo estimé. Cela entraîne que si le tempo est mal estimé, cette probabilité attribuée à chaque note va être faible en moyenne, alors que si le tempo est correctement estimé, la moyenne des probabilités sera maximisée. Nous proposons donc de calculer ces probabilités pour tous les candidats.

Remarque : cette logique ne permet plus de déterminer s’il faut doubler ou diviser le tempo retenu. Il pourra toujours y avoir cette erreur d’un facteur 2 ou bien, on pourrait avoir 2 tempos candidat ex-aequo à la première place.

On choisit donc de garder tous les maximums locaux présents dans la densité de probabilités C. Il y en a obligatoirement au moins 1 mais il peut y en avoir plusieurs, sans limite. On exécute alors la détermination des durées pour chaque tempo τ et on calcule à chaque fois la moyenne µ(τ) des probabilités p des N notes :

On considère alors que le meilleur candidat sera le τ qui maximise µ(τ).

##### Fréquence des durées de notes

Un autre élément à prendre en compte est la lisibilité de la partition. Il est commun en musique d’utiliser majoritairement les durées de notes liées directement par un facteur 2 : les noires, croches et doubles-croches (les blanches et les rondes sont rares). Les durées de notes plus exotiques comme les notes pointées sont plus rares. Il faut donc faire intervenir ce paramètre dans le choix du tempo. De plus, l’écriture d’une partition étant subjective, un musicien préférera noter des croches que des noires (il choisira donc un tempo faible) alors qu’un autre préférera, pour la même partition à transcrire, utiliser des noires. Il aura alors besoin d’un tempo différent. On remarque donc que le choix du tempo dépend également de la proportion de croches (respectivement de noires, de doubles-croches etc…) que l’utilisateur voudra voir figurer dans sa partition.

|  |
| --- |
|  |
| Figure 23 - Heart & Soul écrit avec un tempo respectivement de 120 BPM et 60 BPM |

On crée donc 4 nouveaux paramètres, la proportion de noire, croche, double-croche et autres durées de notes parmi les N notes :

|  |  |
| --- | --- |
|  |  |
|  | 1. 1- |

Où # signifie « nombre de » parmi les N notes du morceau.

Enfin, un cinquième paramètre sert à mettre en évidence une durée parmi toutes les autres. Si l’utilisateur choisi de privilégier l’utilisation des croches par exemple, on calculera de plus :

##### Régression linéaire

Plutôt qu’une SVM, nous choisissons d’utiliser une régression. Nous avons procédé à l’apprentissage des valeurs de pondérations sur les 6 paramètres présentés plus haut ainsi que sur l’unité. La sortie consiste en un scalaire (ou score) spécifique à chaque tempo candidat. On choisira au final, le tempo qui a le plus fort score. Cet apprentissage a été réalisé en utilisant un jeu de données différent de celui que nous avons créé. Il s’agit des jeux de données utilisé pour le concours MIREX. Ces jeux consiste en environ 80 morceaux audio dont le tempo est annoté et correspond à un consensus de plusieurs auditeurs. On retire de ces jeux les morceaux dont les tempos sont annotés en dehors de notre intervalle.

### Résultats obtenus

Notre indicateur permettant d’évaluer la précision de l’estimation de tempo est le suivant :

Cet indicateur est témoin d’une bonne performance lorsqu’il est proche de 0. Si on observe un écart avec un rapport de deux (tempo trouvé = 2x ou 0.5x tempo attendu), l’indicateur vaudra 1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Titre | Tempo attendu (constante) | Tempo trouvé | Rapport tempo |
| 1 | Day Tripper | 140 | 149 | 0,09 |
| 2 | Aller-Retour diatonique | 120 | 59 | 1,02 |
| 3 | Heart & Soul | 60 | 118 | 0,98 |
| 4 | No Surprises | 72 | 72 | 0,00 |
| 5 | Seven Nation Army | 120 | 91 | 0,40 |
| 6 | Hardest Button to Button | 124 | 123 | 0,01 |
| 7 | Johnny B Good | 120 | 132 | 0,14 |
| 8 | Voodoo Child | 96 | 60 | 0,68 |
| 9 | Kashmir | 80 | 81 | 0,02 |
| 10 | Time is running Out | 120 | 123 | 0,04 |
| 11 | 48 notes de la guitare | 90 | 91 | 0,02 |
|  | Moyenne |  |  | 0,30909091 |

Sur notre jeu de donnée, on observe que l’estimation de tempo est très précise (<0.05) pour 5 des 11 morceaux, qu’il y a 2 problème de facteur 2. Pour le morceau 8, « Voodoo Child », l’erreur s’explique par le fait que le morceau soit composé avec plus de croche pointée que de croche ce qui entraine cet écart d’un tiers (1 croche à 90BPM correspond à 1 croche pointée à 120BPM).

Ces résultats, même s’ils sont encore en partie imparfaits sont les meilleurs que nous ayons pu obtenir sur notre jeu de données de manière partiale. L’algorithme pourrait être amélioré de nombreuses façons différentes. Il mériterait également d’être évalué sur un plus grand jeu de donnée.

Comme le tempo est un paramètre important mais qu’il n’est pas toujours bien évalué. Nous avons développé et testé nos algorithmes d’analyse rythmique en utilisant cet algorithme mais également en fixant le tempo à l’avance (étant donné que nous connaissons le tempo réel du morceau).

## Analyse rythmique

Avant toute chose, on précise que l’objectif de cette partie est de détecter au moins 80% de bonnes détections de durées de note.

Cet algorithme s’occupe de la correction de la durée des notes de manière à ce qu’elles puissent être en accord avec un découpage en mesure 4 :4.

Exemple :

Une mesure 4 :4 s’interprète de la manière suivante :

|  |
| --- |
|  |
| Figure 24 - explication de mesure 4:4 |

### Explications

On rappelle que, dons notre application, la durée d’une note est égale au nombre de doubles-croches qui la compose (1 = double-croche, 2 = croche, 3 = croche pointée, 4 = noire, 5 = noire + double-croche) et qu’une mesure 4 :4 comporte 16 doubles-croches.

Lorsque le musicien effectue son enregistrement, il arrive, surtout quand aucun métronome n’est utilisé, qu’il se désynchronise petit à petit du tempo fixé au départ. Ainsi, pour un tempo fixé, une noire au début du morceau n’aura pas la même durée à la fin de l’enregistrement.

Cet algorithme permet ainsi de corriger les erreurs du musicien, par exemple si ce dernier a mal anticipé la durée des notes au sein d’une mesure 4 :4.

Le but ici est donc de normaliser puis de corriger les durées pour que la séquence enregistrée puisse être découpée en multiple de 16.

Description par étape de l’algorithme

Cet algorithme est divisé en deux grandes parties :

* La normalisation des durées
* La correction par multiple de 16

### La normalisation des durées

Une fois le tempo calculé (en battements par minute) et que la liste des onsets dressée (en secondes), nous pouvons construire un vecteur de données correspondant aux durées brutes jouées (en nombre de double-croche) par le musicien de la manière suivante :

|  |
| --- |
| Tempo (bpm)  Onsets/Offsets (secondes)  Durées Brutes (nombre de double-croche)  1,3 2,9 4,4 5,3 |
| Figure 25 - construction du vecteur de durées brutes |

Ensuite, à partir des durées brutes, nous souhaitons avoir un vecteur de durées normalisées. En effet, on ne peut pas avoir au sein d’une partition des notes correspondant à 1.3 ou 2.9 double-croches. C’est pourquoi nous devons passer par une étape de normalisation des durées brutes :

|  |
| --- |
| Onsets/Offsets  Tempo  Durées brutes  1,3  2,9  4,4  5,3  Durées normalisées  1  3  4  6 |
| Figure 26 - construction vecteur durées normalisées |

Pour cela, on va utiliser un peigne de probabilité afin d’avoir en sortie la matrice **out** qui regroupe les durées normalisée (DN) présentant la plus forte probabilité pour une durée en entrée donnée :

|  |
| --- |
|  |
| Figure 27 - Processus de normalisation des durées |

#### Le peigne de probabilités

Un peigne est construit de la manière suivante :

|  |
| --- |
|  |
| Figure 28 - Peigne de gaussiennes |

Chaque courbe correspond à la probabilité pour une note d’être exacte en fonction des durées brutes en entrée.

Afin de construire ces probabilités, on se base sur le fait que certaines notes sont plus souvent présentes que d’autres. Nous avons utilisé les résultats l’étude suivante en les confrontant à nos propres enregistrements pour dresser notre propre distribution d’apparition de note comme suit :

|  |
| --- |
|  |
| Figure 29 - T. Viitaniemi, A Klapuri, « A probabilistic model for the transcription of single-voice melodies », Institute of Signal Processing, Tampere University of Technology, 2003 |

|  |
| --- |
|  |
| Figure 30- Valeurs ajustées |

Voici un exemple d’application de la normalisation pour une durée en entrée (DE) de 4.7:

|  |
| --- |
|  |
| Figure 31 – Normalisation : exemple construction matrice **out** |

On représente ici les probabilités pour la durée 4 (en bleu) et 5 (en vert) pour plus de lisibilité.

On remarque que pour une DE de 4.7, la probabilité d’avoir un 4 est plus élevée que celle d’avoir un 5.  
On construit donc la matrice **out** comme il suit :

|  |  |  |  |
| --- | --- | --- | --- |
| Numéro de note | Durées en Entrée | Durée Normalisée | Proba DN |
| 1 | 4,7 | 4 | 81,9% |

Out

#### Résultats de la normalisation

Voici les résultats en pourcentage de réussite de la normalisation pour les enregistrements que nous avons produits. Nous avons distingué les résultats pour des tempos estimés par l’application de ceux définis directement par l’utilisateur, ce qui nous donne :

|  |  |  |
| --- | --- | --- |
|  | Tempo estimé | Tempo fixé |
| AR diatonique | 0 | 46,66 |
| Heart & soul | 16,21 | 10,81 |
| no surprises | 40 | 40 |
| seven nation | 24,39 | 24,39 |
| hardest | 38,88 | 60,31 |
| johnny B good | 39,06 | 62,5 |
| Voodoo Child | 22,22 | 38,09 |
| Kashmir | 18,66 | 34,84 |
| Time is running | 42,64 | 83,58 |
| 48 notes | 70,08 | 70,08 |
| Moyenne | **45,19** | **56,05** |

Figure 32 - Résultats de la normalisation des notes

On voit donc que cette méthode fonctionne bien car on se retrouve avec une moyenne pondérée à hauteur de 45% pour un tempo estimé et de 56% pour un tempo fixé par l’utilisateur.

En revanche, comme précisé au début de cette partie, on souhaite avoir un pourcentage de bonne détection de l’ordre de 80%. Nous avons donc besoin d’une seconde étape afin d’essayer d’augmenter ce pourcentage. On passe alors par l’étape suivante : la correction des durées.

### Correction des durées et découpage en mesures régulières

Le but de la correction par multiple de 16 est de modifier certaines durées normalisées afin d’avoir des notes qui puissent rentrer dans une mesure 4 :4.

|  |
| --- |
| Durées normalisées  1  3  4  6  Correction & Découpage en mesure |
| Figure 33 - processus de correction des durées |

Cela nous permet, après avoir normalisé les durées, de passer d’une série de notes anarchiques et ne possédant aucune signification musicale à une partition découper en mesure cohérente et étant le plus proche possible de l’enregistrement d’origine.

Si on reprend l’exemple utilisé dans la normalisation (DE = 4.7) :

|  |
| --- |
|  |
| Figure 34 - Correction : exemple construction matrice **out** |

Cette fois, nous allons garder les deux durées qui entourent la durée brute en entrée comme suit :

4 < 4.7 < 5

Ce qui nous donne la matrice **out** suivante :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Numéro de note | Durées en Entrée (DE) | Durée Inférieure (DI) | Proba DI | Durée Supérieure (DS) | Proba DS |
| 1 | 4,7 | 4 | 81,9% | 5 | 15,4% |

out

On change donc cette matrice en conservant la durée inférieure et supérieure à la note en entrée ainsi que leur probabilité respective.

C’est ici que commence l’algorithme de correction des durées.

Le but sera d’utiliser la matrice **out** pour construire la matrice **mesures**.

La matrice **mesures** permet de donner une idée de la composition de chaque mesure créée par l’algorithme comme ci-dessous :

|  |
| --- |
|  |
| Figure 35 - Correction : exemple construction matrice **mesures** |

Dans notre exemple, on se retrouve donc avec deux mesures : la première comporte quatre noires, la seconde est composée de deux noires suivies de quatre croches, ce qui est cohérent avec le découpage en mesure 4 :4, soit 16 double-croches par mesure.

Pour en arriver à ce résultat, on utilise 3 paramètres dans cet algorithme :

* La matrice **mesures** comme on l’a vu plus tôt, classant les notes en mesures :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Note 1 | Note 2 | Note … | Note n |
| Mesure 1 |  |  |  |  |
| Mesure 2 |  |  |  |  |
| Mesure … |  |  |  |  |
| Mesure n |  |  |  |  |

* La matrice **mesureTemporaire** , un buffer qui va permettre de gérer la mesure actuelle si celle-ci nécessite d’être corrigé par l’algorithme.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Note 1 | Note 2 | Note … | Note n |
| Durée avec la plus forte proba (D1) |  |  |  |  |
| Proba D1 |  |  |  |  |
| Durée avec la plus faible proba (D2) |  |  |  |  |
| Proba D2 |  |  |  |  |
| Certitude |  |  |  |  |

Cette matrice reprend les informations de la matrice **out** pour les classer ainsi :

* D1 sera la durée possédant la plus forte probabilité entre DI et DS
* D2 sera la plus faible
* La certitude est un paramètre calculé ainsi : certitude = proba D1 – proba D2. Ce paramètre s’est révélé efficace pour détecter quelles notes seraient prioritaires au cours du processus de correction.
* Le paramètre **somme** qui donne simplement la somme des durées au sein de la mesure en cours de traitement.

Afin maintenant d’expliquer le fonctionnement de l’algorithme, nous allons utiliser un exemple concret d’un véritable enregistrement

#### Exemple : Voodoo Child – Jimmy Hendrix (2 premières mesures)

Pour cet exemple, on commence après la génération de la matrice **out** suivante :

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Numéro de note | Durées en Entrée (DE) | Durée Inférieure (DI) | Proba DI | Durée Supérieure (DS) | Proba DS |
| 1 | 2,7 | 2 | 53,6% | 3 | 43,6% |
| 2 | 0,9 | 1 | 94,4% | 2 | 5,7% |
| 3 | 1,9 | 2 | 99% | 1 | 1% |
| 4 | 2,0 | 2 | 99% | 3 | 1% |
| 5 | 2,0 | 2 | 99% | 3 | 1% |
| 6 | 1,7 | 2 | 98,9% | 1 | 1,1% |
| 7 | 0,9 | 1 | 94,3% | 2 | 5,7% |
| 8 | 1,1 | 1 | 77,2% | 2 | 22,7% |
| 9 | 2,0 | 2 | 99% | 3 | 1% |
| 10 | 2,7 | 3 | 59% | 2 | 37,8% |
| 11 | 1,1 | 1 | 86,3% | 2 | 13,7% |
| 12 | 2,2 | 2 | 98,9% | 3 | 1,1% |
| 13 | 3,8 | 4 | 99% | 3 | 1% |
| 14 | 7,1 | 6 | 74% | 8 | 26% |

Figure 36 - Voodoo Child : **out**

Et voici la matrice **mesures** que l’on est censé avoir en sortie de l’algorithme :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Mes. 1 | 3 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 |
| Mes. 2 | 3 | 1 | 2 | 4 | 6 | 0 | 0 | 0 | 0 |

Figure 37 - Voodoo Child : **mesures** attendues

La correction est constituée de 3 étapes successives :

1. Le remplissage des différentes matrices
2. Le dépassement de mesure
3. Gérer les mesures incomplètes

##### Etape 1 : remplissage des matrices

On commence tout d’abord par comparer deux à deux les probabilités successivement pour chaque DE afin de retenir celles ayant la plus forte à partir de la matrice out de la manière suivante :

|  |
| --- |
|  |
| Figure 38- Etape 1 : Schéma |

Le terme somme nous indique la somme des durées de la mesure en cours.

Dans notre exemple, cela donne pour la première mesure :

**Mesures :**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Mes. 1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 3 | 2 |

**mesureTemporaire :**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| D1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 3 |
| P(D1) | 53,6% | 94,4% | 99% | 99% | 99% | 98,9% | 94,3% | 77,2% | 99% | 59% |
| D2 | 3 | 2 | 1 | 3 | 3 | 1 | 2 | 2 | 3 | 2 |
| P(D2) | 43,6% | 5,7% | 1% | 1% | 1% | 1,1% | 5,7% | 22,7% | 1% | 37,8% |
| Cert. | 9,9% | 88,5% | 98% | 98% | 98% | 97,8% | 88,5% | 54,5% | 98% | 21,2% |

Ici, le terme **somme = 17**, on passe donc à l’étape suivante.

##### Etape 2 : dépassement de mesure

|  |
| --- |
|  |
| Figure 39 - Etape 2 : Explications Schéma |

Ici nous allons détailler les blocs « changement de la durée », « gestion dernière note » et « procédure d’échec de correction ».

Gestion dernière note : ce bloc sauvegarde dans un vecteur la dernière colonne de la matrice **mesureTemporaire** pour ensuite la supprimer, ainsi que la dernière case de la matrice **mesures**. Nous aurons l’occasion de l’expliquer au cours de l’exemple.

Changement de la durée : ce bloc échange la durée D1 par D2 si D1 < D2. En effet, comme nous sommes dans la partie qui concerne les dépassements de mesure, le fait de changer D1 par D2 si D2 > D1 n’a pas de sens. Il faut noter que si un changement est effectué, la certitude de la note passe à 100. De ce fait, si l’algorithme remarque que toutes les certitudes sont à leur maximum, il enclenchera la « procédure d’échec de correction ».

Procédure d’échec de correction : ce bloc s’exécute une fois que l’algorithme aura essayé de corriger toutes les durées sans pour autant arriver atteindre **somme** = 16. A ce moment, la dernière note est réintégrée dans la mesure courante et un signal est généré pour indiquer que cette mesure est fausse et que le programme n’a pas été en mesure de corriger le problème.

Nous allons maintenant commencer à illustrer cela grâce à l’exemple.

|  |
| --- |
|  |
| Figure 40 - Etape 2 : Schéma 1 |

On commence par vérifier qu’on est dans le cas d’un dépassement de mesure. Comme on l’a fait remarquer plus haut, **somme** = 17, on peut donc avancer dans l’algorithme et chercher la note présentant la certitude la plus faible. Dans notre cas, il s’agit de la note 1 :

**Mesures :**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | 9 | 10 |
| Mes. 1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 3 | 2 |

**mesureTemporaire** :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| D1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 3 |
| P(D1) | 53,6% | 94,4% | 99% | 99% | 99% | 98,9% | 94,3% | 77,2% | 99% | 59% |
| D2 | 3 | 2 | 1 | 3 | 3 | 1 | 2 | 2 | 3 | 2 |
| P(D2) | 43,6% | 5,7% | 1% | 1% | 1% | 1,1% | 5,7% | 22,7% | 1% | 37,8% |
| Cert. | 9,9% | 88,5% | 98% | 98% | 98% | 97,8% | 88,5% | 54,5% | 98% | 21,2% |

**Somme = 17**

|  |
| --- |
|  |
| Figure 41 - Etape 2 : Schéma 2 |

On remarque que D2 est supérieur à D1. On arrive donc à la partie « gestion de la dernière note » de l’algorithme.

**Mesures :**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | 9 | 10 |
| Mes. 1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 3 | 2 |

**mesureTemporaire :**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| D1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 3 |
| P(D1) | 53,6% | 94,4% | 99% | 99% | 99% | 98,9% | 94,3% | 77,2% | 99% | 59% |
| D2 | 3 | 2 | 1 | 3 | 3 | 1 | 2 | 2 | 3 | 2 |
| P(D2) | 43,6% | 5,7% | 1% | 1% | 1% | 1,1% | 5,7% | 22,7% | 1% | 37,8% |
| Cert. | 9,9% | 88,5% | 98% | 98% | 98% | 97,8% | 88,5% | 54,5% | 98% | 21,2% |

**Somme = 17**

A ce moment, la dernière note est sauvegardée dans un vecteur temporaire et est ensuite supprimée de la matrice **mesureTemporaire** et remplacée par un zéro dans la ma matrice **mesures**. De cette manière, on la réintègre pour la mesure suivante, où elle sera placée en première position.

|  |
| --- |
|  |
| Figure 42 - Etape 2 : Schéma 3 |

Ce qui nous donne :

**Mesures :**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | 9 | 10 |
| Mes. 1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 3 | 0 |

**mesureTemporaire :**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| D1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 |
| P(D1) | 53,6% | 94,4% | 99% | 99% | 99% | 98,9% | 94,3% | 77,2% | 99% |
| D2 | 3 | 2 | 1 | 3 | 3 | 1 | 2 | 2 | 3 |
| P(D2) | 43,6% | 5,7% | 1% | 1% | 1% | 1,1% | 5,7% | 22,7% | 1% |
| Cert. | 9,9% | 88,5% | 98% | 98% | 98% | 97,8% | 88,5% | 54,5% | 98% |

**Somme = 17**

|  |
| --- |
|  |
| Figure 43 - Etape 2 : Schéma 4 |

Après avoir fait cette manipulation, le programme vérifie si toutes les certitudes ne sont pas égales à 100. N’ayant pas passé par le bloc « changement de la durée », aucune certitude n’a bougé.

**Mesures :**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | 9 | 10 |
| Mes. 1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 3 | 0 |

**mesureTemporaire :**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| D1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 |
| P(D1) | 53,6% | 94,4% | 99% | 99% | 99% | 98,9% | 94,3% | 77,2% | 99% |
| D2 | 3 | 2 | 1 | 3 | 3 | 1 | 2 | 2 | 3 |
| P(D2) | 43,6% | 5,7% | 1% | 1% | 1% | 1,1% | 5,7% | 22,7% | 1% |
| Cert. | 9,9% | 88,5% | 98% | 98% | 98% | 97,8% | 88,5% | 54,5% | 98% |

**Somme = 17**

|  |
| --- |
|  |
| Figure 44 - Etape 2 : Schéma 5 |

On procède donc à une nouvelle vérification de la somme.

Maintenant, **somme** = 15, nous sortons donc de l’algorithme pour passer à l’étape suivante.

**Mesures :**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | 9 | 10 |
| Mes. 1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 3 | 0 |

**mesureTemporaire :**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| D1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 |
| P(D1) | 53,6% | 94,4% | 99% | 99% | 99% | 98,9% | 94,3% | 77,2% | 99% |
| D2 | 3 | 2 | 1 | 3 | 3 | 1 | 2 | 2 | 3 |
| P(D2) | 43,6% | 5,7% | 1% | 1% | 1% | 1,1% | 5,7% | 22,7% | 1% |
| Cert. | 9,9% | 88,5% | 98% | 98% | 98% | 97,8% | 88,5% | 54,5% | 98% |

**Somme = 15**

##### Etape 3 : mesure incomplète

Cette étape s’effectue suivant l’algorithme ci-dessous :



Figure 45 - Etape 3 : Schéma 1

On commence par vérifier que la mesure nécessite une correction. En effet, si à l’issue de l’étape 2, **somme** = 16, l’étape 3 n’est pas nécessaire. Nous cherchons la certitude la plus faible. Comme l’étape 2 n’a pas modifié les certitudes, c’est à nouveau la note 1.

**Mesures :**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | 9 | 10 |
| Mes. 1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 3 | 0 |

**mesureTemporaire :**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| D1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 |
| P(D1) | 53,6% | 94,4% | 99% | 99% | 99% | 98,9% | 94,3% | 77,2% | 99% |
| D2 | 3 | 2 | 1 | 3 | 3 | 1 | 2 | 2 | 3 |
| P(D2) | 43,6% | 5,7% | 1% | 1% | 1% | 1,1% | 5,7% | 22,7% | 1% |
| Cert. | 9,9% | 88,5% | 98% | 98% | 98% | 97,8% | 88,5% | 54,5% | 98% |

**Somme = 15**



Figure 46 - Etape 3 : Schéma 2

Cette fois-ci, contrairement à l’étape 2, nous cherchons, ici dans l’étape 3, non pas à diminuer les durées des notes mais à les augmenter pour atteindre **somme** = 16. Nous ne changerons donc que les notes où D2 > D1, ce qui est le cas de la note 1.

Si on avait eu D2 < D1, nous aurions quand même passé la certitude de la note à 1, afin de notifier que celle-ci a été traitée mais non modifiée.

**Mesures :**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | 9 | 10 |
| Mes. 1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 3 | 0 |

**mesureTemporaire :**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| D1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 |
| P(D1) | 53,6% | 94,4% | 99% | 99% | 99% | 98,9% | 94,3% | 77,2% | 99% |
| D2 | 3 | 2 | 1 | 3 | 3 | 1 | 2 | 2 | 3 |
| P(D2) | 43,6% | 5,7% | 1% | 1% | 1% | 1,1% | 5,7% | 22,7% | 1% |
| Cert. | 9,9% | 88,5% | 98% | 98% | 98% | 97,8% | 88,5% | 54,5% | 98% |

**Somme = 15**



Figure 47 - Etape 3 : Schéma 3

Ce qui nous donne :

**Mesures :**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | 9 | 10 |
| Mes. 1 | 3 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 3 | 0 |

**mesureTemporaire :**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| D1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 |
| P(D1) | 53,6% | 94,4% | 99% | 99% | 99% | 98,9% | 94,3% | 77,2% | 99% |
| D2 | 3 | 2 | 1 | 3 | 3 | 1 | 2 | 2 | 3 |
| P(D2) | 43,6% | 5,7% | 1% | 1% | 1% | 1,1% | 5,7% | 22,7% | 1% |
| Cert. | 100% | 88,5% | 98% | 98% | 98% | 97,8% | 88,5% | 54,5% | 98% |

**Somme = 15**

La durée est changée dans la matrice mesures et la certitude passe à 100.



Figure 48 - Etape 3 : Schéma 4

Après une nouvelle vérification des certitudes, nous ne sommes pas dans le cas d’une « procédure d’échec de correction » et nous retournons au début de l’algorithme.

**Mesures :**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | 9 | 10 |
| Mes. 1 | 3 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 3 | 0 |

**mesureTemporaire :**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| D1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 |
| P(D1) | 53,6% | 94,4% | 99% | 99% | 99% | 98,9% | 94,3% | 77,2% | 99% |
| D2 | 3 | 2 | 1 | 3 | 3 | 1 | 2 | 2 | 3 |
| P(D2) | 43,6% | 5,7% | 1% | 1% | 1% | 1,1% | 5,7% | 22,7% | 1% |
| Cert. | 100% | 88,5% | 98% | 98% | 98% | 97,8% | 88,5% | 54,5% | 98% |

**Somme = 15**



Figure 49 - Etape 3 : Schéma 5

On recalcule une nouvelle fois la somme, et on se rend compte que nous arrivons à **somme** = 16, ce qui nous fait sortir de l’algorithme.

**Mesures :**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 5 | 6 | 7 | 8 | 9 | 10 |
| Mes. 1 | 3 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 | 3 | 0 |

**mesureTemporaire :**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| D1 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 |
| P(D1) | 53,6% | 94,4% | 99% | 99% | 99% | 98,9% | 94,3% | 77,2% | 99% |
| D2 | 3 | 2 | 1 | 3 | 3 | 1 | 2 | 2 | 3 |
| P(D2) | 43,6% | 5,7% | 1% | 1% | 1% | 1,1% | 5,7% | 22,7% | 1% |
| Cert. | 100% | 88,5% | 98% | 98% | 98% | 97,8% | 88,5% | 54,5% | 98% |

**Somme = 16**

Nous allons traiter la seconde mesure de la même manière pour finalement récupérer la matrice **mesures** suivante :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Mes. 1 | 3 | 1 | 2 | 2 | 2 | 2 | 1 | 1 | 2 |
| Mes. 2 | 3 | 1 | 2 | 4 | 6 | 0 | 0 | 0 | 0 |

Figure 50 - Voodoo Child : matrice **mesures** en sortie

Ce qui correspond à la matrice **mesures** désirée au début de l’exemple.

#### Résultats de la correction

Voici les résultats en pourcentage de réussite de la normalisation en incluant la correction pour les enregistrements que nous avons produits. Encore une fois, nous distinguons les résultats pour les tempos estimés et fixés par l’utilisateur. Afin de bien voir l’apport de la correction, on rappelle dans les colonnes « no correction » les résultats de la normalisation sans l’étape de correction.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Tempo estimé** | | **Tempo fixé** | |
| **no correction** | **correction** | **no correction** | **correction** |
| **AR diatonique** | 0 | 20 | 46,66 | 100 |
| **Heart & soul** | 16,21 | 14,70 | 10,81 | 100 |
| **no surprises** | 40 | 92,30 | 40 | 92,30 |
| **seven nation** | 24,39 | 37,5 | 24,39 | 72,5 |
| **Hardest button** | 38,88 | 81,69 | 60,31 | 81,69 |
| **johnny B good** | 39,06 | 70,49 | 62,5 | 76,19 |
| **Voodoo Child** | 22,22 | 30 | 38,09 | 51,25 |
| **Kashmir** | 18,66 | 81,94 | 34,84 | 82,19 |
| **Time is running** | 42,64 | 83,58 | 83,58 | 85,07 |
| **48 notes** | 70,08 | 97,8 | 70,08 | 97,8 |
| **MOYENNE** | **45,19** | **79,70** | **56,06** | **89,32** |

Figure 51 - Résultats de la correction des notes

On voit bien que la correction améliore grandement les résultats de bonnes détections de durées par rapport à la normalisation seule en passant de 45% à 80% de bonnes détections pour un tempo estimé par le logiciel et de 56% à 89% de bonnes détections pour un tempo fixé par l’utilisateur.

Les résultats arrivent donc à la hauteur des attentes que nous avions fixées au début du projet.

### Améliorations

L’algorithme nécessite cependant quelques améliorations pour être encore plus fonctionnel. Notamment :

* Gérer les mesures fausses : Dans l’état actuel de l’application, la « procédure d’échec de correction » indique seulement les mesures où la correction n’a pas fonctionné. Il faudrait probablement une quatrième étape pour tenter de gérer au mieux ce cas de figure, notamment au niveau de la resynchronisation des notes en cas de fausses mesures.
* Gérer les autres découpages de mesure : pour le moment, l’application ne corrige que les partitions en 4 :4. Une amélioration serait de gérer les autres découpages possibles (4 :3,

## Analyse Harmonique

Cette partie intervient à la suite de la segmentation des onsets. Elle consiste à déterminer la note et l’octave de l’onset extrait (correspondant à une seule note). Il existe de nombreuses méthodes différentes que nous n’avons pas toutes implémentées et sur lesquelles il serait intéressant de travailler. Nous avons donc listé les principales méthodes, puis comme pour l’onset detection, nous avons essayé au fil de notre projet plusieurs méthodes, en essayant de les combiner, ou d’inventer une méthode par une approche intuitive.

Au fil des recherches que nous avons pu faire, nous avons constaté que les études portant sur la reconnaissance de parole englobent quasiment toutes les recherches sur la détection de la fréquence fondamentale.

### Approches temporelles

Ces méthodes consistent à regarder le signal d'entrée comme une amplitude fluctuante dans le domaine temporel et essayer de trouver des motifs de répétition dans la forme d'onde qui donnent des indices quant à sa périodicité. La forme d’onde qui représente cette modification de la pression dans l’air est alors analysée pour pouvoir dégager la fréquence fondamentale.

#### Zero crossing method

Une technique simple qui consiste à compter le nombre de fois que le signal croise le niveau de référence 0 par unité de temps. Si la puissance spectrale de la forme d'onde est concentrée autour de f0, alors il va franchir la ligne zéro deux fois par cycle, comme sur la figure 1. Cependant, si la forme d'onde contient des composantes spectrales de fréquences plus élevées, comme dans la figure 2, alors le signal pourrait franchir la ligne zéro plus de deux fois par cycle. Une solution est de mettre un filtre qui atténue ces hautes fréquences qui détériorent la détection, mais la fréquence de coupure du filtre doit être bien choisie pour ne pas supprimer une fréquence fondamentale f0 qui se situerai dans les hautes fréquences.

|  |
| --- |
| /Users/apple/Desktop/Capture d’écran 2016-02-02 à 09.44.31.png |
| Figure 52 - Méthode zero crossing |

#### Peak rate

Cette méthode compte le nombre de pics positifs par seconde dans le signal. En théorie, le signal aura une valeur maximale et minimale pour chaque cycle. On n’aurait qu’a compter ces valeurs maximales pour déterminer la fréquence du signal. Dans la pratique, un détecteur de crête locale doit être utilisé pour trouver l'endroit où la forme d'onde est localement le plus grand et le nombre de ces maxima locaux en une seconde est la fréquence de la forme d'onde, à moins que chaque période de la forme d'onde contient plus d'un maximum local.

**Inconvénient de ces méthodes :** Nous récupérons un signal audio de guitare comprenant normalement une seule note. Cependant, du fait du timbre de l’instrument, d’autres fréquences font leurs apparitions dans la représentation fréquentielle de la transformée de Fourier. Les méthodes décrient ci-dessus fonctionne plutôt bien sur des sons très purs (sans modifications), mais dans le cadre de notre projet, la guitare émet un spectre fréquentiel trop complexe pour que ces méthodes donnent de bons résultats.

#### YIN estimator

La difficulté avec les techniques d'auto-corrélation réside dans le fait que les pics se produisent à des sous-harmonique, donc il est parfois difficile de déterminer quel pic représente la fréquence fondamentale et quels pics font références aux harmoniques et autres fréquences contenues dans le signal.

YIN tente de résoudre ces problèmes en plusieurs manières. Cet algorithme est basé sur la fonction de différence, qui, bien que similaire à l’autocorrélation, tente de minimiser la différence entre le signal et le signal retardé-décalé, au lieu de maximiser le produit (autocorrélation).

#### Convolution

La méthode proposée ici est de générer une banque de sinus correspondant aux fréquences fondamentales que l’on souhaite détecter. On procède ensuite à un produit de convolution entre le signal x et l’ensemble des signaux constituants la banque de fréquences (les sinus et x doivent avoir la même longueur).

Le produit de convolution exprime la quantité de recouvrement d’une fonction y lorsqu’on la déplace sur une autre fonction x. Ainsi, le sinus qui possède la quantité de recouvrement la plus grande avec x (le sinus qui ressemblera le plus à notre signal) sera considérée comme celui qui se rapproche le plus de notre signal, et sa fréquence sera alors la fréquence fondamentale.

#### Autocorrelation

La corrélation entre deux formes d'onde est une mesure de la similitude. Les formes d'onde sont comparées à intervalles de temps différents, et leur "identité" est calculé à chaque intervalle. Le résultat d'une corrélation est une mesure de similarité en fonction du décalage dans le temps entre les débuts des deux formes d'onde.

Pour des signaux périodiques, une caractéristique d'auto-corrélation intéressante est que la fonction d'auto-corrélation est elle-même périodique. Comme le décalage augmente à la moitié de la période de la forme d'onde, la corrélation atteint un minimum. En effet, la forme d'onde est hors de phase avec sa copie retardée dans le temps. Comme le décalage augmente à nouveau à la longueur d'une période, l'auto-corrélation atteint un maximum, car la forme d'onde et sa copie retardée dans le temps sont en phase. Le premier pic de l'auto-corrélation indique la période de la forme d'onde (lorsque les harmoniques ne sont pas trop présentes).

Certains problèmes surviennent avec cette méthode, notamment quand on fait l’autocorrélation d’un signal pseudoperiodique de nature harmonique complexe (figure 2).

Le premier pic ne correspond pas forcément à la période de la forme d'onde complète, mais à la période d’une des harmoniques (ou d’une autre fréquence composant le timbre de l’instrument) du signal. La difficulté est alors de pouvoir distinguer entre "grands" et "petits" pics, selon que les harmoniques et autres fréquences sont plus ou moins présentes.

|  |
| --- |
|  |
| Figure 53 - Signal et résultat de l'autocorrélation du signal |

### Approches fréquentielles

Dans ce mode de fonctionnement, nous avons utilisé pour certaines méthodes la STFT (Short Time Fourier Transform) qui est une transformée de Fourier particulière pour déterminer la fréquence sinusoïdale et la phase d'une section locale d'un signal.

Donc pour chaque segment d’une chanson (un segment correspondant à un onset), on applique la STFT.

On en déduit la fréquence fondamentale par les méthodes présentées ci-dessous. Grâce à cette fréquence, on peut en déduire la hauteur de la note et donc son octave, en veillant à ce que ces fréquences soient comprises dans le spectre fréquentiel de la guitare (on recherche grâce à une fonction la valeur de la fréquence la plus proche de celle trouvée correspondant à une fréquence de la guitare).

#### Harmonic product spectrum (HPS)

La fréquence fondamentale peut être déterminée en mesurant les fréquences des composantes harmoniques supérieures et en calculant le plus grand dénominateur commun de ces fréquences harmoniques.

Le plus grand dénominateur commun peut être déterminé en découpant le spectre.

Pour réaliser le découpage, il suffit de diminuer la fréquence d'échantillonnage du spectre par 2,3,4,5 etc.. à partir du premier échantillon. Ensuite on recherche le pic le plus élevé dans le résultat de la multiplication des ces spectres pour retrouver la fréquence fondamentale (les pics se décalent).

Cette méthode est peu coûteuse en temps de calcul, résistante au bruit (additif ou multiplicatif), réglable à différents types d'entrées (en changeant le nombre d'harmoniques à envisager ou à remplacer les multiplications par des additions).

|  |
| --- |
| Capture%20d’écran%202016-02-02%20à%2014.51.41.png |
| Figure 54 - Harmonic product spectrum |

#### Méthodes intuitives

L’idée était de prendre les pics les plus importants grâce à un seuil et de les étudier. Nous avons remarqué que sur certains morceaux, les spectres fréquentiels correspondant aux notes jouées présentaient comme plus fort pic la fréquence de l’harmonique inférieure. Les fréquences de la note fondamentale, de la tierce ainsi que la quinte étaient également présentes de façon régulière.

Un fois ces pics isolés, nous pouvions alors déterminer la fréquence fondamentale. Cependant, ce schéma ne se présentait pas toujours et parfois la fréquence fondamentale était le plus fort pic sur la transformée de Fourier locale. En prenant ces changements en compte, cet algorithme fut performant mais que sur certains morceaux. Il n’est malheureusement pas adaptable dans la majorité des morceaux que nous avons testés.

Une autre méthode intuitive, assez similaire à la première, était de prendre les pics de plus forte amplitude et de regarder la somme des amplitudes ayant les mêmes fréquences multiples. Ainsi nous aurions pu dégager les fréquences harmoniques de la note la plus présente et déterminer la note jouée. Cependant, il se trouve que cette règle n’est pas toujours respectée. En effet, la fréquence (et ses multiples) la plus présente dans le spectre fréquentiel n’est pas toujours celle de la fréquence fondamentale dans le cas de notre guitare.

Nous avons également essayé de combiner ces deux méthodes mais les résultats finaux n’étaient pas aboutissants. Il serait intéressant de tester ces méthodes sur des instruments ayant des timbres spécifiques et différents de celui de notre guitare.

|  |
| --- |
|  |
| Figure 55 - Transformée de Fourier rapide du Mi-3 de la guitare |

Dans cet exemple de STFT, on voit bien que la note Mi-3 de la guitare possède un spectre fréquentiel composé : du Mi-3 pour le premier pic, du Mi-4 pour le second pic, de la tierce pour le 3ème pic et enfin du Mi-5 pour le quatrième.

#### Cepstral method

La transformée de Fourier d’un signal a un nombre régulier de pics, représentant le spectre harmonique. Lorsque l’on prend le logarithme du spectre, les amplitudes des pics changent d’échelle. On obtient alors une forme d’onde périodique dans le domaine fréquentiel, où la période du signal est liée à la fréquence fondamentale du signal d’origine. Une transformation de Fourier inverse de ce signal possède alors un pic représentant la fréquence recherchée.

|  |
| --- |
|  |
| Figure 56 - Cesptral method |

On constate que le signal est bien périodique Avec la présence des pics à égal distance dans le spectre

Le cepstre est une transformation de ce signal du domaine temporel vers un autre domaine analogue au domaine temporel.

La formule correspondant à ce nouveau domaine temporel est :



### Combinaison de méthodes

Après avoir codé et testé la plupart de ces méthodes, les meilleurs résultats sont obtenus par la méthode de convolution pour la reconnaissance des tons (notes). L’autre méthode retenue pour la reconnaissance des octaves est l’Harmonic product spectrum.

### Résultats

Pour finir, nous obtenons comme résultats de bonnes détections : Tons: 93,08

Octaves: 92,7%

|  |
| --- |
|  |
| Figure 57 - Evolution des résultats de reconnaissance des tons et octaves au long du projet |

Les différents pics représentent les différentes méthodes testées. La baisse à certains niveaux peut aussi correspondre à un ajout de données pour les tests (nouveaux morceaux dans la base de données).

# Création d‘un logiciel à destination de l’utilisateur

Dans le but de conclure notre projet et de prouver la performance de notre système d’une autre façon, nous avons décidé de proposer un logiciel utilisable par un musicien lambda, c’est-à-dire en proposant de réaliser toute la manipulation sur une seule plateforme, de l’enregistrement à la visualisation de la partition, sans passer par le logiciel Matlab qui reste une interface pour initiés.

Comme nous n’avions pas le temps, et que cela aurait été s’écarter du sujet, de développer un logiciel d’enregistrement ainsi qu’un logiciel de visualisation de partition de musique, nous avons préféré utiliser des outils déjà existants. Ce sont les mêmes outils que nous utilisions pour enregistrer un morceau de notre jeu de données et pour visualiser et écouter la sortie de notre algorithme. Nous utilisons donc le logiciel Audacity pour l’enregistrement et le logiciel Guitar Pro pour la visualisation du fichier MIDI sous forme de partition. Si Audacity est un logiciel libre et gratuit, ce n’est pas le cas de Guitar Pro. C’est pourquoi, à la première utilisation de notre logiciel, si l’utilisateur ne possède pas ces logiciels, il peut en indiquer un autre.

Concrètement, notre logiciel se présente sous la forme d’une interface avec 4 étapes :

|  |
| --- |
|  |
| Figure 58 - Capture d'écran de l'interface utilisateur de notre logiciel |

## Étape 1 : Enregistrement

Dans cette première étape nous proposons à l’utilisateur de créer un nouveau projet Audacity (bouton « New »), ou d’en ouvrir un existant (bouton « Explore »). Cela a pour implication d’ouvrir le logiciel Audacity avec le projet en question. Avec ce logiciel, l’utilisateur peut enregistrer, monter, modifier son instrument comme il le veut. Il faut ensuite qu’il utilise la fonctionnalité d’export au format WAVE d’audacity. Si l’utilisateur à fermer le logiciel Audacity, il peut rouvrir le dernier projet utilisé grâce au bouton « Open ».

## Étape 2 : Choix du fichier audio

Dans cette étape, l’utilisateur doit choisir un fichier audio WAVE existant via le bouton « Load » qui ouvre un explorateur. Il peut s’agir du fichier que l’utilisateur vient de créer à l’étape précédente, ou d’un autre fichier audio. Quand il a choisi un fichier, le bouton « Play » s’active et permet de jouer le fichier audio dans l’application par défaut (VLC, Windows Media Player, etc… selon les paramètres de l’utilisateur).

## Étape 3 : Paramétrage de la transcription

À cette étape, l’utilisateur peut choisir le format d’export. À la fin de notre projet, la génération au format GP4 (Guitar Pro 4) n’était pas encore fonctionnelle. L’utilisateur peut également imposer un tempo s’il le connait dans l’intervalle 55 BPM / 180 BPM. Il faut cocher la case adjacente pour activer cette fonctionnalité. Autrement, le logiciel détectera automatiquement le tempo. Le cadre « Found tempo » sera rempli avec la valeur estimé du tempo après la transcription.

## Étape 4 : Choix du fichier de sortie et génération

Enfin, l’utilisateur doit donner un nom à son fichier de sortie. Il peut également choisir le dossier d’enregistrement via le bouton « … ». L’appui sur le bouton « Save » lance la génération du fichier de sortie avec une barre d’avancement qui s’affiche en bas. Après cette génération, le logiciel Guitar Pro (ou autre selon l’utilisateur) s’ouvre avec le nouveau fichier.

## Utilisation de notre algorithme

Le développement de notre projet s’est fait entièrement en langage Matlab. Lorsque nous avons développé ce logiciel, nous avons voulu porter notre code en dll, exe ou librairie C/C++ de façon à pouvoir être utilisé de façon indépendante de l’environnement Matlab. Cependant nous n’avons pas réussi immédiatement à faire cela et le manque de temps nous oblige à rester dépendants de cet environnement. Le logiciel final ne peut donc, dans l’absolu, pas être diffuser à des utilisateurs qui n’aurait pas Matlab d’installer sur leur machine.

Notre solution se limite donc à lancer notre algorithme complet dans une instance Matlab cachée de façon à ce que l’utilisateur n’ait pas à interagir avec Matlab directement.

|  |
| --- |
|  |
| Figure 59 - Capture d'écran de l'interface utilisateur de notre logiciel en cours de génération |

# Améliorations possibles

# Conclusion

Le développement de notre projet a été un succès. Nous avons réussi à atteindre les objectifs que nous nous étions fixés, et ce dans le temps imparti pour le projet. Il s'agissait d'un projet très motivant dans un domaine original qui nous a séduit. Nous avons tous largement étendu nos compétences techniques, notamment dans l'implémentation d'algorithme sophistiqués, issus de publications scientifiques. Le système obtenu à la fin du projet répond largement à ce que l'on pouvait en attendre, et bien qu'il n'en soit pas au stade d'une diffusion publique, nous pensons à l'utiliser à titre personnel dans son état actuel.

D'un point de vue de la méthode, nous avons mis un point d'honneur à commencer le projet en implémentant les méthodes qui nous ont permis de l'évaluer. Cela nous a permis d'avoir un critère objectif nous permettant de décider l'adoption d'une nouvelle méthode ou pas. La limite de cette façon de faire est que puisque les résultats d'une étape peuvent avoir une forte influence sur d'autre. Ainsi une légère amélioration sur l'onset detection pouvait améliorer beaucoup plus les résultats de l'analyse harmonique, qu'un changement majeur d'algorithme d'analyse harmonique. D'autre part, le choix des morceaux que nous avons ajouté dans notre jeu de données était purement subjectif et soumis à nos goûts ou idées du moment. Il aurait certainement mieux fallu commencer par enregistrer des morceaux simples, et apporter au fur et à mesure du projet de la complexité dans la transcription des morceaux.

# Table des figures

[Figure 1 - Schéma fonctionnel général de notre système de transcription automatique 3](#_Toc444098401)

[Figure 2 - Exemple de confrontation entre les onsets annotés et les onsets détectés 6](#_Toc444098402)

[Figure 3 - Exemple d'une matrice de confusion de la détermination des durées (AR) 7](#_Toc444098403)

[Figure 4 - Exemple de matrices de confusions de la détermination des notes (AH) 8](#_Toc444098404)

[Figure 5 - Onset 10](#_Toc444098405)

[Figure 6 - Algorithme de détection d'onset 11](#_Toc444098406)

[Figure 7 - Spectrogramme d'un morceau de 25 secondes 12](#_Toc444098407)

[Figure 8 - Simplification de la distance euclidiènne 15](#_Toc444098408)

[Figure 9 - Comparaison de deux méthodes (Spectral flux & Complex domain) 17](#_Toc444098409)

[Figure 10 - Comparaison de deux méthodes (Phase deviation & Complex domain) 18](#_Toc444098410)

[Figure 11 - Combinaison de méthodes (Spectral flux & Complex domain) 18](#_Toc444098411)

[Figure 12 - Exemple de détection d'onset sur "Hardest Button to Button" 19](#_Toc444098412)

[Figure 13 - Algorithme final d'onset detection 20](#_Toc444098413)

[Figure 14 - Evolution des résultats de la détection d'onset 20](#_Toc444098414)

[Figure 15 - Comparaison des pourcentages de détection d'onset par méthodes 21](#_Toc444098415)

[Figure 16 - Puissance d'onset des 10 premières secondes de "No Surprises" 23](#_Toc444098416)

[Figure 17 - Autocorrélation généralisée sur la première fenêtre de "No Surprises" 24](#_Toc444098417)

[Figure 18 - Autocorrélation aux harmoniques augmentées de la 1ère fenêtre de "No Surprises" 24](#_Toc444098418)

[Figure 19 - Construction des trains d'impulsions 25](#_Toc444098419)

[Figure 20 - Probabilité qu'un tempo soit représentatif d'une fenêtre 26](#_Toc444098420)

[Figure 21 - Probabilité qu'un tempo soit représentatif du morceau 26](#_Toc444098421)

[Figure 22 - Exemple de mise en échec de la méthode proposée 28](#_Toc444098422)

[Figure 23 - Heart & Soul écrit avec un tempo respectivement de 120 BPM et 60 BPM 29](#_Toc444098423)

[Figure 24 - explication de mesure 4:4 31](#_Toc444098424)

[Figure 25 - construction du vecteur de durées brutes 32](#_Toc444098425)

[Figure 26 - construction vecteur durées normalisées 32](#_Toc444098426)

[Figure 27 - Processus de normalisation des durées 33](#_Toc444098427)

[Figure 28 - Peigne de gaussiennes 34](#_Toc444098428)

[Figure 29 - T. Viitaniemi, A Klapuri, « A probabilistic model for the transcription of single-voice melodies », Institute of Signal Processing, Tampere University of Technology, 2003 34](#_Toc444098429)

[Figure 30- Valeurs ajustées 35](#_Toc444098430)

[Figure 31 – Normalisation : exemple construction matrice **out** 35](#_Toc444098431)

[Figure 32 - Résultats de la normalisation des notes 36](#_Toc444098432)

[Figure 33 - processus de correction des durées 37](#_Toc444098433)

[Figure 34 - Correction : exemple construction matrice **out** 37](#_Toc444098434)

[Figure 35 - Correction : exemple construction matrice **mesures** 39](#_Toc444098435)

[Figure 36 - Voodoo Child : **out** 41](#_Toc444098436)

[Figure 37 - Voodoo Child : **mesures** attendues 41](#_Toc444098437)

[Figure 38- Etape 1 : Schéma 42](#_Toc444098438)

[Figure 39 - Etape 2 : Explications Schéma 43](#_Toc444098439)

[Figure 40 - Etape 2 : Schéma 1 44](#_Toc444098440)

[Figure 41 - Etape 2 : Schéma 2 45](#_Toc444098441)

[Figure 42 - Etape 2 : Schéma 3 46](#_Toc444098442)

[Figure 43 - Etape 2 : Schéma 4 47](#_Toc444098443)

[Figure 44 - Etape 2 : Schéma 5 48](#_Toc444098444)

[Figure 45 - Etape 3 : Schéma 1 49](#_Toc444098445)

[Figure 46 - Etape 3 : Schéma 2 50](#_Toc444098446)

[Figure 47 - Etape 3 : Schéma 3 52](#_Toc444098447)

[Figure 48 - Etape 3 : Schéma 4 53](#_Toc444098448)

[Figure 49 - Etape 3 : Schéma 5 54](#_Toc444098449)

[Figure 50 - Voodoo Child : matrice **mesures** en sortie 55](#_Toc444098450)

[Figure 51 - Résultats de la correction des notes 55](#_Toc444098451)

[Figure 52 - Méthode zero crossing 57](#_Toc444098452)

[Figure 53 - Signal et résultat de l'autocorrélation du signal 58](#_Toc444098453)

[Figure 54 - Harmonic product spectrum 59](#_Toc444098454)

[Figure 55 - Transformée de Fourier rapide du Mi-3 de la guitare 60](#_Toc444098455)

[Figure 56 - Cesptral method 61](#_Toc444098456)

[Figure 57 - Evolution des résultats de reconnaissance des tons et octaves au long du projet 62](#_Toc444098457)

[Figure 58 - Capture d'écran de l'interface utilisateur de notre logiciel 63](#_Toc444098458)

[Figure 59 - Capture d'écran de l'interface utilisateur de notre logiciel en cours de génération 64](#_Toc444098459)

# Références

|  |  |
| --- | --- |
| [1] | D. Gerhard, «Pitch Extraction and Fundamental Frequency: History and Current Techniques,» University of Regina, 2003. |
| [2] | C. D. M. D. M. S. Juan P. Bello, «On the Use of Phase and Energy for Musical Onset Detection in the Complex Domain,» *IEEE SIGNAL PROCESSING LETTERS,* vol. 11, n° %16, 2004. |
| [3] | S. Dixon, «Simple Spectrum-Based Onset Detection,» Austrian Research Institute for Artificial Intelligence Freyung, 2010. |
| [4] | L. D. S. A. C. D. M. D. M. B. S. Juan Pablo Bello, «A Tutorial on Onset Detection in Music Signals,» *IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING,* vol. 13, n° %15, 2005. |
| [5] | S. Dixon, «Onset detection revisited,» Austrian Research Institute for Artificial Intelligence Freyung, 2006. |
| [6] | E. Benetos et Y. STYLIANOU, «Auditory spectrum-based pitched instrument onset detection,» *Audio, Speech, and Language Processing,* vol. 18, n° %18, 2010. |
| [7] | W. F. Crawford Tait, «Wavelet analysis for onset detection,» Department of computing science, University of Glasgow, G12 8QQ, 1996. |
| [8] | C. CHING-HUA, «Audio onset detection using machine learning techniques: the effect and applicability of key and tempo information,» Department of Computer Science, University of Southern California Viterbi School of Engineering, Los Angeles, California, USA, 2008. |
| [9] | G. Percival, «Streamlined Tempo Estimation Based on Autocorrelation and Cross-Correlation With Pulses,» *IEE/ACM Transactions on Audio, Speech, And Language Processing,* vol. 22, 2014. |
| [10] | J. P. B. M. D. M. S. Chris Duxbury, «Complex domain onset detection for musical signals,» Department of Electronic Engineering Queen Mary, University of London Mile End Road, London E1 4NS, UK, 2003. |
| [11] | Wikipedia, «Article sur l’onset detection,» [En ligne]. Available: https://en.wikipedia.org/wiki/Onset\_%28audio%29. |
| [12] | R. R. D. M. d. M. Carlos Ros, «Influence of peak selection methods on onset detection,» ISCTE-IUL L2F/INESC-ID Lisboa, 2012. |
| [13] | L. D. G. R. Pierre LEVEAU, «Methodology and tools for the evaluation of automatic onset detection algorithms in music,» Laboratoire d’Acoustique Musicale, 2004. |
| [14] | P. d. l. Cuadra, «PITCH DETECTION METHODS REVIEW,» [En ligne]. Available: https://ccrma.stanford.edu/~pdelac/154/m154paper.htm. |
| [15] | A. De Cheveigne et H. Kawahara, «YIN, a fundamental frequency estimator for speech and music,» Ircam-CNRS, 1 place Igor Stravinsky, 75004 Paris, France, 2002. |
| [16] | D. Gerhard, «Techniques, Pitch Extraction and Fundamental Frequency: History and Current,» 2003. |
| [17] | A. Von dem Knesebeck et U. Zölzer, «Comparison of pitch trackers for real-time guitar effects,» Dept. of Signal Processing and Communications, Helmut Schmidt University Hamburg, Germany, 2010. |
| [18] | J. Di Martino et Y. Laprie, «An Efficient F0 Determination Algorithm Based on the Implicit Calculation of the Autocorrelation of the Temporal Exciation Signal,» 1999. |
| [19] | E. Benetos et S. Dixon, «Joint Multi-pitch Detection using Harmonic envelope Estimation for Polyphonic Music,» 2011. |
| [20] | A. Klapuri, «Sound onset detection by applying psychoacoustic knowledge,» Signal Processing Laboratory, Tampere University of Technology P.O.Box 553, FIN-33101 Tampere, FINLAND, 1999. |
| [21] | A. Klapuri, «Multipitch Analysis of Polyphonic Music and Speech Signals Using an Auditory Model,» *IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING,* vol. 16, n° %12, 2008. |