Sommaire

[1. Introduction 1](#_Toc443652196)

[2. Contexte et objectifs 1](#_Toc443652197)

[2.1. Contexte 1](#_Toc443652198)

[2.2. Objectifs et limites posées 2](#_Toc443652199)

[3. Méthode d’évaluation 2](#_Toc443652200)

[3.1. Critères d’évaluation 2](#_Toc443652201)

[3.2. Base de données annotée 2](#_Toc443652202)

[4. Détection d’onset 2](#_Toc443652203)

[5. Estimation du tempo 2](#_Toc443652204)

[5.1. Estimation de tempo par fenêtrage, autocorrélation et corrélation avec des trains d’impulsions. 3](#_Toc443652205)

[5.2. Résultats obtenus 4](#_Toc443652206)

[6. Analyse de la composition rythmique 4](#_Toc443652207)

[7. Correction des durées et découpage en mesures régulières 4](#_Toc443652208)

[8. Analyse Harmonique 4](#_Toc443652209)

[9. Création d‘un logiciel à destination de l’utilisateur 4](#_Toc443652210)

[10. Améliorations possibles 4](#_Toc443652211)

[11. Conclusion 4](#_Toc443652212)

# 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 - 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.

# Détection d’onset

# 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 [2]. 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 »* [2], 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 5 - 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 6 - 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 7 - Autocorrélation aux harmoniques augmentées de la première 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 8 - 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 - 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 - 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 - 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 12 - 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 de la composition rythmique

# Correction des durées et découpage en mesures régulières

# Analyse Harmonique

# 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 13 - 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 14 - Capture d'écran de l'interface utilisateur de notre logiciel en cours de génération |

# Améliorations possibles

# Conclusion

# Table des figures

[Figure 1 - Schéma fonctionnel général de notre système de transcription automatique 3](#_Toc443664519)

[Figure 2 - Exemple de confrontation entre les onsets annotés et les onsets détectés 5](#_Toc443664520)

[Figure 3 - Exemple d'une matrice de confusion de la détermination des durées (AR) 7](#_Toc443664521)

[Figure 4 - Exemple de matrices de confusions de la détermination des notes (AH) 8](#_Toc443664522)

[Figure 5 - Puissance d'onset des 10 premières secondes de "No Surprises" 10](#_Toc443664523)

[Figure 6 - Autocorrélation généralisée sur la première fenêtre de "No Surprises" 11](#_Toc443664524)

[Figure 7 - Autocorrélation aux harmoniques augmentées de la première fenêtre de "No Surprises" 12](#_Toc443664525)

[Figure 8 - Construction des trains d'impulsions 13](#_Toc443664526)

[Figure 9 - Probabilité qu'un tempo soit représentatif d'une fenêtre 14](#_Toc443664527)

[Figure 10 - Probabilité qu'un tempo soit représentatif du morceau 14](#_Toc443664528)

[Figure 11 - Exemple de mise en échec de la méthode proposée 16](#_Toc443664529)

[Figure 12 - Heart & Soul écrit avec un tempo respectivement de 120 BPM et 60 BPM 17](#_Toc443664530)

[Figure 13 - Capture d'écran de l'interface utilisateur de notre logiciel 19](#_Toc443664531)

[Figure 14 - Capture d'écran de l'interface utilisateur de notre logiciel en cours de génération 21](#_Toc443664532)

# Références

|  |  |
| --- | --- |
| [1] | D. Gerhard, «Pitch Extraction and Fundamental Frequency: History and Current Techniques,» University of Regina, 2003. |
| [2] | 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. |