# Estimation de 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. 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 »*, que nous appelerons 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 1 - 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

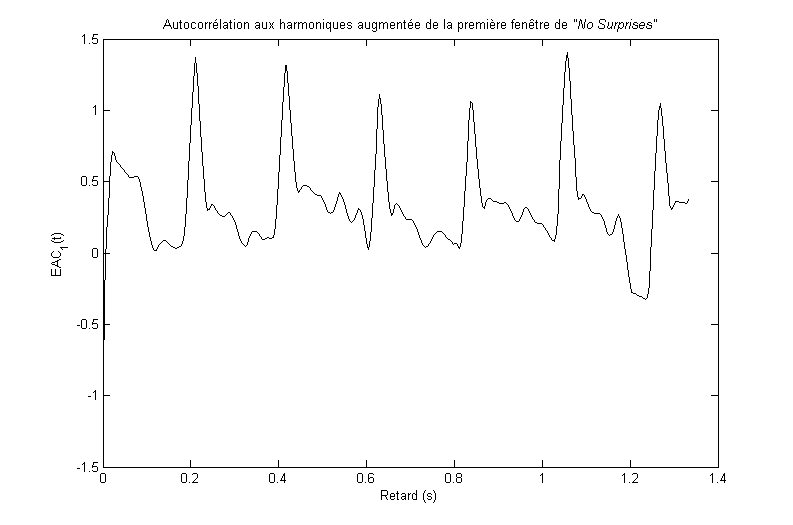
Pour chaque fenêtre obtenue, on calcul ensuite l’autocorrélation généralisée. Ce calcul se fait en passant par le domaine fréquentiel (par FFT), en compressant le spectre obtenu par un coefficient *c*, puis à revenir dans le domaine temporelle (IFFT) :

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 2 - 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 :



### 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 3 - 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 une 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 4 - 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 5 - 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)

1. La probabilité que le tempo soit inférieur à L :
2. La probabilité que le tempo soit supérieur à L
3. La probabilité que le tempo soit proche de L
4. La probabilité que le tempo soir proche de L/2 (rapport de ½ )
5. La probabilité que le tempo soir proche de 2L (rapport de 2 )
6. La somme de ces 3 dernières probabilités
7. Les rapports entre L et le deuxième meilleur candidat
8. Le rapport entre L et le 3ème meilleur candidat
9. Le nombre de tempo pour lequel la densité de probabilité n’est pas nulle (en mode discret)
10. 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.

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 6 - 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 7 - 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 | DayTripper | 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 | Voodo 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, « Voodo 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).