ENSICAEN

ÉCOLE NATIONALE SUPÉRIEURE D'INGÉNIEURS DE CAEN
& CENTRE DE RECHERCHE
Spécialité Informatique
2ème année

Détection de rythme dans amaroK

MASSÉ Nicolas
LIMIN Thomas

Détection de rythme dans amaroK

Remerciements:

M Luc BRUN, qui nous a suivis pendant ce projet et qui nous a fournis la documentation nécessaire pour démarrer

M Jérome Lebosse, pour avoir partager avec nous ses précédents travaux.

M Miguel Alonso, pour ses explications concernant la SEF et pour nous avoir donné accès aux implémentation de ses travaux.

1 - Introduction

Pour ce projet de seconde année de cycle ingénieur préparé à l'ENSICAEN, nous avons choisi le sujet proposé par M. Luc BRUN intitulé: « Participation à un projet SourceForge » et décrit comme suit: « Les étudiants sont invités à choisir un projet sur le site de SourceForge (http://sourceforge.net/index.php) Il faut pour cela choisir un projet qui vous intéresse et choisir dans la liste des TODO une ou plusieurs tâches à réaliser ».

En effet, nous sommes depuis déjà quelques années de fervents utilisateurs de logiciels libres, profitant au maximum des libertés que nous garantis la licence sous laquelle sont distribués ces logiciels. Cependant, nous ne sommes la plupart du temps, si ce n'est tout le temps, que de simple utilisateurs, profitant du travail d'autrui sans contreparties aucune. Fort de nos compétences grandissantes en développement logiciel, nous avons décidés de saisir l'occasion qui nous était donnée d'appliquer la notion de partage chère au logiciel libre et de nous impliquer, cette fois en tant que développeurs.

Notre choix s'est porté sur le projet amaroK (« Rediscover Your Music! »), hébergé à l'url suivante: http://sourceforge.net/projects/amarok. Selon la description fournie sur la page d'accueil, amarok est un lecteur de musique pour GNU/Linux et Unix possédant une interface graphique intuitive qui permet de jouer plus facilement que jamais la musique que l'on aime, tout en le faisant « joliment ». Plus prosaïquement, c'est un logiciel profondément intégré à KDE, un des environnements de bureau les plus utilisés sur les postes de travail GNU/Linux, et implémentant une gestion très efficace de la collection de fichiers musicaux et des listes de lecture. Ces deux caractéristiques en font un outil à l'utilisation simple et efficace.

La liste des améliorations proposées et des fonctionnalités à implémenter comporte, entre autres, une proposition concernant l'ajout d'une fonction de détection automatique du rythme:

Implement beat detection (thread?), interface should glow/move to the beat, visualisations have access to beat/bpm info.

C'est à cette partie du développement d'amaroK que nous avons choisi de participer, ceci pour plusieurs raison:

- Une majorité des propositions s'articulent principalement autour de la correction de dysfonctionnements. Le choix d'une de ces options comporte le désavantage d'avoir à analyser de longues portions de code source et à en comprendre précisément le fonctionnement pour en débusquer les erreurs. Tâche fastidieuse que l'auteur du code en question sera beaucoup plus à même de réaliser plus efficacement que de nouveaux venus.
- Un classement des propositions par priorité existant, nous avons préféré, compte tenu du planning associé au projet, nous limiter à celles classées dans la catégorie « long terme ».
- L'implémentation de la détection du rythme, l'option retenue, fait appel à des notions avancées de traitement du signal, un sujet qui a été abordé en première année, ce qui nous a permis d'en appréhender les contours. Malheureusement, nous n'avons pas du tout, durant notre formation initiale (DUT Informatique), appris à maîtriser les outils mathématiques au programme des enseignements de

classes préparatoires qui ont été à la base de la compréhension de cette matière. Cet handicap nous a été extrêmement préjudiciable, et notre maîtrise des filtres et autres transformées temps/fréquence s'en est fortement ressentie. C'est cependant un domaine de l'informatique qui nous est apparu comme étant aussi fondamental qu'intéressant, c'est pourquoi nous avons mis à profit le projet de seconde année pour approfondir personnellement ces questions.

Malgré le sujet choisi et l'intérêt que nous portons à amaroK, ce rapport s'articulera exclusivement autour de la problématique de la détection du rythme. En effet, les difficultés rencontrées au cours de l'avancée du projet nous ont contraint, faute de ressources temporelles disponibles en quantité suffisante, à réduire le sujet à l'unique étude d'un algorithme de détection de rythme efficace et à son l'implémentation. Nous présenterons donc tout d'abord les fondements d'un tel algorithme ainsi que les outils informatiques et mathématiques nécessaires. Ensuite seront analysés et comparés les différentes solutions proposées puis enfin, les possibles voies de poursuite de notre travail.

Sommaire

2 - Détection du rythme

2.1 - Vue d'ensemble

2.1.1 - Un réflexe naturel

Une réaction fréquente d'un individu à l'écoute d'un morceau de musique est de taper du pied, de claquer ses doigts ou de faire tout autre geste susceptible de marquer le tempo du son qu'il perçoit. Cette attitude est très naturelle en conséquence de quoi le processus cognitif qui est à sa source semble très simple: tout un chacun peut réaliser ce traitement alors même qu'il est concentré sur une toute autre tâche. Cependant, dès que l'on s'attache à reproduire ce phénomène de manière automatique, à l'aide d'un programme ou d'une machine, on s'aperçoit qu'il n'en est rien.

Les mécanismes permettant au cerveau humain d'extraire un rythme à partir d'un extrait musical ne sont pas précisément connus, mais le consensus autour du tempo est admis. Il suffit d'observer un groupe de danseurs pour s'apercevoir que tous, hormis quelques énergumènes qui peuvent se distinguer par leur taux d'alcoolémie ou leur besoin d'être remarqué, tous suivent la même cadence. Il est possible d'argumenter en faveur d'un comportement de meute, chacun suivant l'exemple d'un possible meneur, mais dans ce cas, ces danseurs feraient ils différemment s'il étaient seuls, utiliseraient-ils un autre tempo? Assurément non, c'est pourquoi on peut s'accorder sur le fait qu'il existe des critères permettant de définir un rythme et que l'on peut travailler sur une méthode automatique pour le déterminer.

2.1.2 - Pourquoi détecter le rythme?

Pourquoi détecter le rythme? Voilà effectivement un point qui ne saurait être négligé. Toute réponse correcte découlant d'une question bien formulée, il est indispensable de savoir dans quelle situation la détection du rythme peut s'avérer utile pour correctement prendre en compte les contraintes qui sont en jeux.

Comme indiqué dans le TODO d'amaroK, la détection du rythme peut être utilisée pour cadencer des animations qui peuvent ainsi servir de « fond visuel » (par glissement de l'expression « fond sonore »). Mais ce n'est pas tout. Nous avons listé différentes applications, qui peuvent être séparée en deux groupes: les applications temps réel et les applications en temps différé.

Applications en temps différé:

- Principalement le calcul d'un tempo (nommé BPM: beat per minute) moyen calculé globalement ou sur un sous ensemble suffisamment représentatif d'un morceau musical, afin de l'utiliser comme une caractéristique, à l'instar du genre ou du nom de l'auteur, permettant un classement par tempo. C'est une utilisation probable dans le cadre d'amaroK, qui utilise déjà intensivement les autre critères de classement dans ce but. Cette application n'a pas la contrainte du temps réel, en effet le calcul du tempo peut s'effectuer une fois pour toute en tâche de fond pendant l'indexation (par exemple) de la collection et le résultat stocké pour de multiples références ultérieures.

Les applications temps réel:

- Amélioration du « cross-fading » qui permet une transition agréable entre deux morceau grâce à une variation du volume (baisse pour celui qui se termine, hausse pour celui qui commence). La détection instantanée du rythme peut permettre un léger ajustement de la vitesse de lecture pour amoindrir ou éliminer une désagréable impression de déphasage. C'est ce que font, à l'oreille, les disc-jockey pour enchaîner les pistes.
- Génération d'événements dont la périodicité est couplée avec celle d'un son, pour des animation, des plug-in de visualisation, des jeux de lumière.

Dans le cas des application temps réel, la contrainte temporelle reposant sur l'algorithme est très forte, son temps de calcul doit être plus faible que la durée de l'extrait analysé, et ceci même si les performance de la machine support sont faibles.

De plus, dans les deux cas, le traitement doit pouvoir être effectué quelque soit le type musical de l'extrait (bien que certain type de musique électronique apparaissent comme étant beaucoup plus facile à analyser que de la musique classique).

2.1.3 - Points de départ de l'analyse

La détection du rythme est généralement effectuée à l'aide de deux étapes distinctes: la détection des attaques (onset en anglais), c'est à dire les brusques variation des propriétés du signal, comme par exemple les changements de notes, complété par une analyse des dates d'apparition de ces attaques pour en déterminer la périodicité et ainsi trouver le tempo de l'extrait musical.

2.2 - Outils utilisés

Bien que l'outil majoritairement utilisé dans le domaine de l'automatique et du traitement de signal soit le logiciel Matlab développé par la société Mathworks, nous avons préféré l'utilisation de Scilab, developpé par l'INRIA (Institut national de la recherche en informatique et automatique, www.inria.fr). En effet, ne disposant pas d'une expérience importante de Matlab (moins de 6h), ni du droit de l'utiliser (seulement 15 jours d'évaluation gratuite ou 99$ pour la version étudiant), nous avons recherché des logiciels alternatif et trouver deux représentant: Scilab (http://www.scilab.org) et Octave (http://www.gnu.org/software/octave/). Après un rapide test il est apparu que Scilab, convenait mieux à nos besoins, principalement axés autour des fonctions de traitement du signal. Les implémentations effectuées pour ce projet sont donc des scripts pour Scilab. Leur syntaxe est proche de celle des scripts Matlab, mais elle est incompatibles, ce qui nous a contraint à traduire les script qui nous ont été fournis.

Une implémentation en C des algorithmes testés via les scripts Scilab est prévue, car les performances des script est assez faible et limite l'étude à de court extrait d'au maximum 15s. Une implémentation en C pourrait permettre l'analyse d'extraits plus long, voire une analyse en temps réel.

3 - Algorithmes de détection de rythme

Avant d'élaborer un algorithme de détection de rythme, il convient de définir ce qu'est un battement. Il en existe plusieurs définition, mais dans un premier temps, nous considérerons qu'« un battement est une hausse brutale de l'énergie du flux sonore ». Cette définition est à la base de l'algorithme suivant.

3.1 - Analyse statistique du flux sonore

Le système auditif humain détermine le rythme de la musique en détectant la périodicité d'une suite battements. Le signal sonore acquis par l'oreille contient une certaine énergie qui est ensuite convertie en un signal « électrique » que le cerveau interprète. Plus un son est fort, plus il transporte de l'énergie. Cependant un son ne sera perçu comme un battement que si son énergie est largement supérieure à l'énergie des sons précédents. Le cerveau détecte alors une variation brutale dans l'énergie du flux sonore. C'est pourquoi si l'oreille capte un son monotone avec de temps en temps de fortes hausses d'énergie, le cerveau percevra un battement. À l'opposé, si elle capte un fort son monotone, elle ne percevra pas de pulsation. Une pulsation est donc une brusque hausse de l'énergie.

Dans cet algorithme, on détecte les variations d'énergie en calculant l'énergie moyenne du signal, et en la comparant à son énergie instantanée. Cette dernière se calcule en effectuant la moyenne de 1024 échantillons (ce qui représente environ 5 centièmes de secondes pour une fréquence d'échantillonnage de 44100 Hz). L'énergie moyenne du signal ne doit pas être calculée sur l'ensemble de la chanson car il peut y avoir des passages calmes et d'autres plus intenses. L'énergie instantanée doit être comparée à l'énergie moyenne environnante afin qu'une fin de chanson très énergique n'influence pas le début, possiblement plus calme. Pour cela, on calcule une moyenne glissante des énergies instantanées. Le nombre d'échantillons pris en compte par cette moyenne doit représenter la persistance énergétique du système auditif humain, c'est-à-dire le temps mis par celui-ci pour « oublier » le son qu'il vient d'entendre. Il a été déterminé que ce temps vaut environ une seconde (soit 43 échantillons d'énergie instantanée). C'est un compromis entre une trop grande durée qui induit une influence inappropriée d'énergies trop lointaines pour être prises en compte par le système auditif et une durée trop courte qui aurait pour effet de ne détecter que des très courtes variations d'énergie.

On définit le flux sonore par un vecteur x d'échantillons de longueur N, l'énergie instantanée par un vecteur E de longueur $n = N / 1024$, et l'énergie moyenne par un vecteur El également de longueur n. On a alors les équations suivantes :

$$E(i)=\begin{cases}\sum_{k=1024\times i}^{1024\times(i+1)} x(k)^2 & \text{si } i\in[0;n-1]\\ 0 & \text{sinon}\end{cases}$$

$$\forall i\in[0;n-1], E_l(i)=\frac{1}{43}\sum_{k=i}^{i+43} E(k)$$

Ensuite, pour chaque élément de E, on le compare à C x El, où C est une constante déterminant la sensibilité de l'algorithme (en général 1.3). Si E est supérieur, on a une pulsation.

La figure 1 présente les résultats obtenus sur quinze secondes de « Brainwasher » de Daft Punk. La figure supérieure montre les échantillons du signal (c'est à dire la « forme » du signal) ainsi que les pulsations détectées par l'algorithme (lignes verticales rouges). La figure inférieure montre l'énergie instantanée (courbe verte, avec de fortes variations) et l'énergie moyenne (courbe bleue, quasi constante). On remarque que les pulsations détectées sont trop nombreuses et irrégulières (ce signal en comporte en fait 30 régulièrement espacés). On peut expliquer cette sur-sensibilité par la forme bruitée de la courbe de l'énergie instantanée.

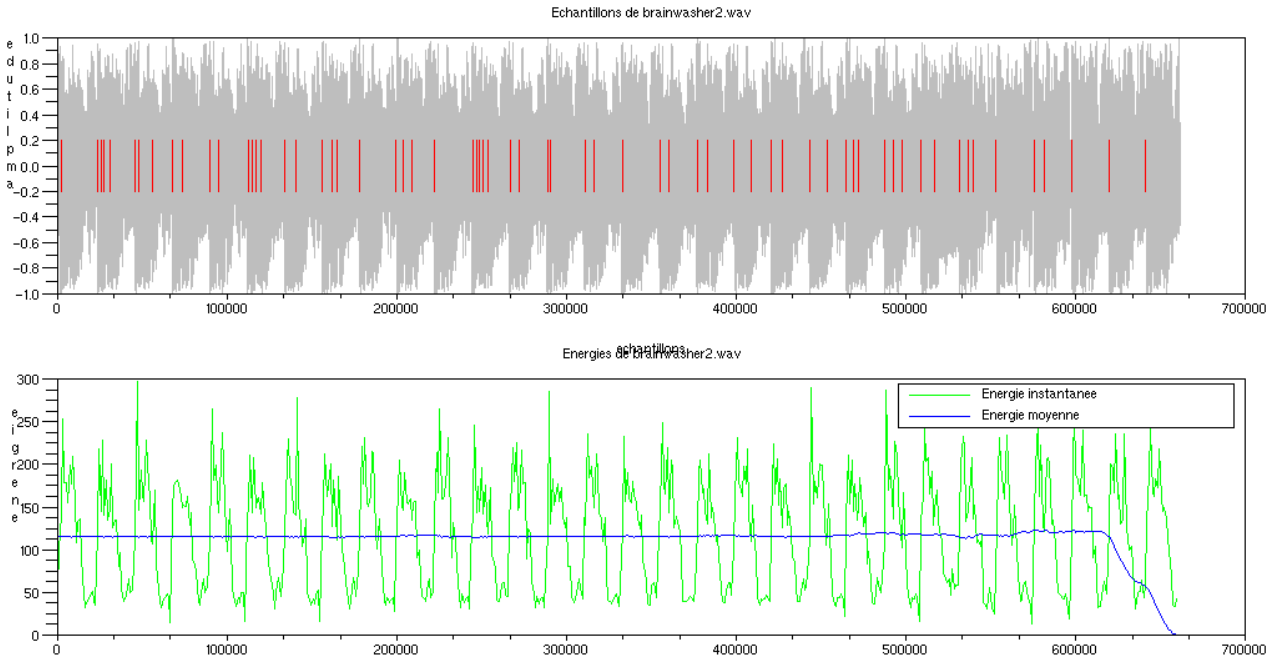

Illustration 1: Détection de rythme par analyse statistique du flux

Pour corriger cela, on peut lisser la courbe grâce à une moyenne glissante sur cinq éléments, on obtient alors le résultat de la figure 2, pour plus de lisibilité, les deux courbes d'énergie ont été remplacées par la différence des deux (E - C x El, avec C = 1.3). La figure supérieure montre la forme du signal, ainsi que les pulsations détectées, grâce à la différence non lissée (traits supérieurs bleus) et grâce à la différence lissée (traits inférieurs rouges). Les deux figures suivantes montrent la courbe de la différence non lissée (bleue) et lissée (rouge).

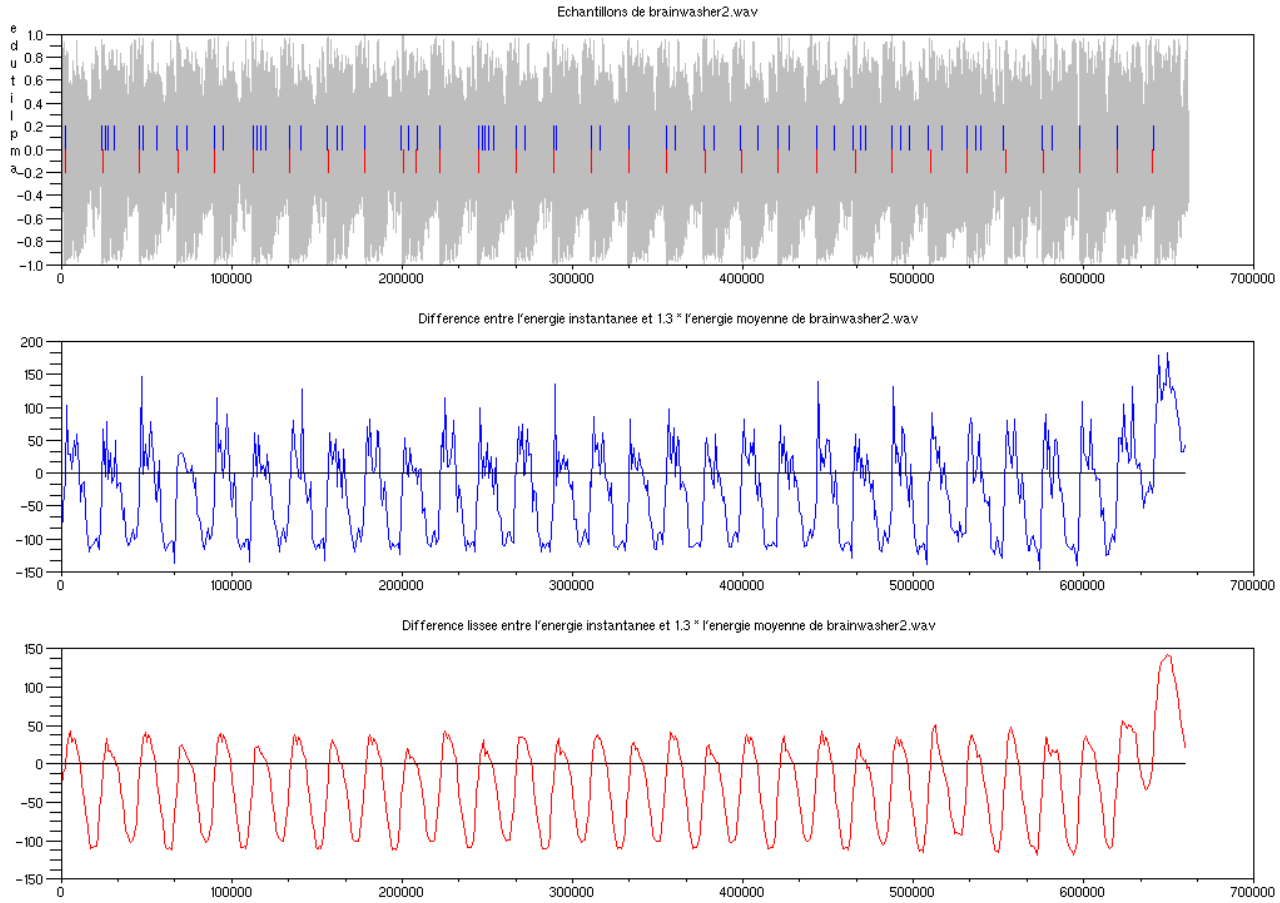

Illustration 2: Amélioration de la détection par lissage de courbe

On remarque une très nette amélioration de la détection avec la courbe lissée par la moyenne glissante. On note, cependant, qu'avec un autre échantillon légèrement plus compliqué (les quinze première secondes de « Satisfaction » de Benny Benassy), voir figure 3, l'algorithme est mis en défaut.

En effet, les pulsations obtenues par lissage de la courbe de différence ne détecte pas toutes les pulsations et celles détectées ne sont pas périodiques (le signal comprends 31 pulsations régulièrement espacées). Après observation de la courbe de différence, on peut remarquer que : sur le premier tiers du signal, si la courbe était déplacée vers le haut, on détecterait des pulsations plus régulières, sur le deuxième tier, c'est l'inverse et le dernier tier est correct. En partant de ces constatations, il est possible d'améliorer l'algorithme. En effet, en faisant varier C, on peut modifier la position de la courbe par rapport à l'axe des abcisses.

Le choix de la constante C dépend du type de musique analysée. Par exemple, pour de la techno ou du rap, où les pulsations sont très marquées on choisira C = 1.4, alors que pour des musiques plus « bruitées » comme du rock ou du hard rock on choisira C = 1.1 ou C = 1.0. L'idéal serait de laisser l'algorithme choisir la valeur de C. Pour cela on calcule la variance des énergies instantanées, ce qui montrera si les pulsations sont marquées où non.

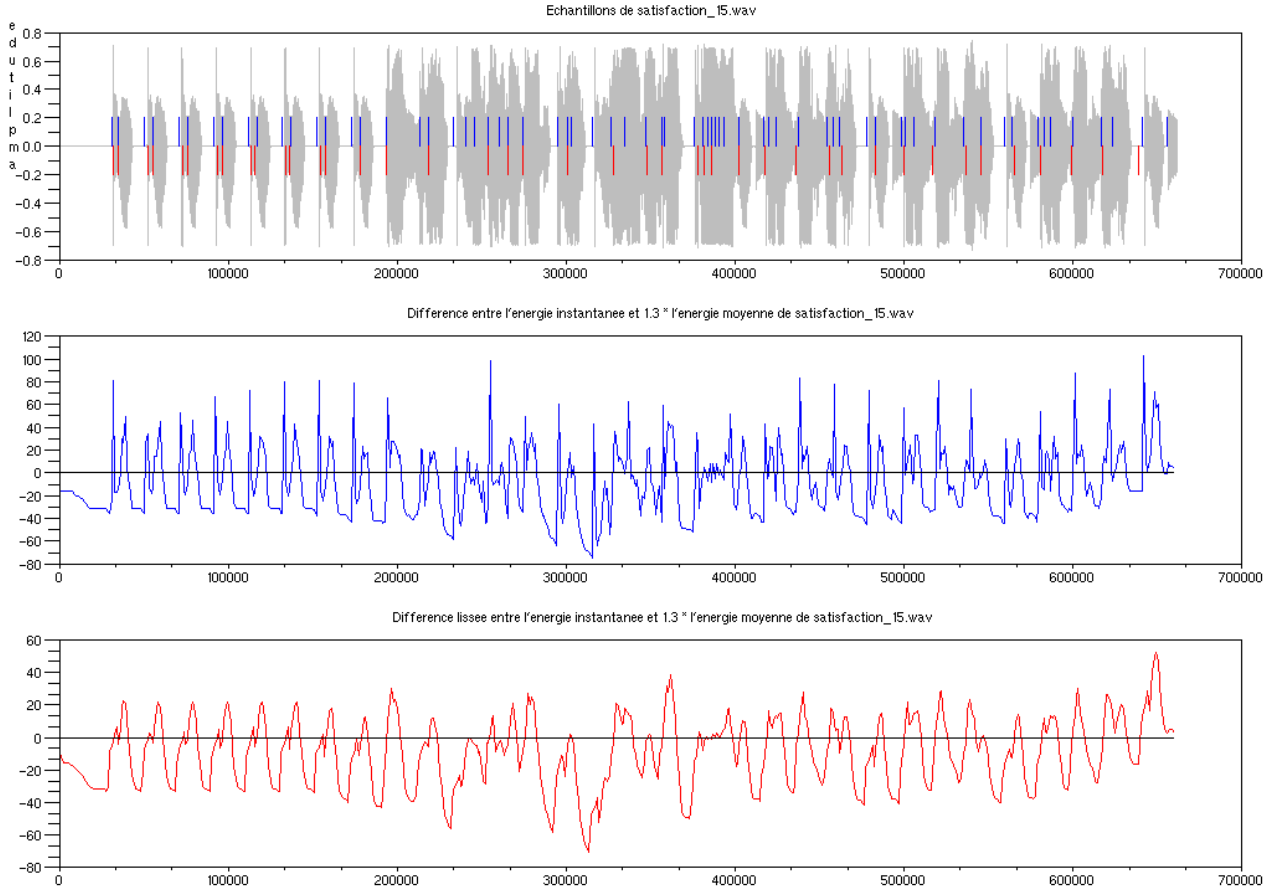

Illustration 3: L'algorithme est mis en défaut

On calcule la variance V de la manière suivante :

$$\forall i \in [0\,;n-1], V(i) = \frac{1}{43}\sum_{k=i}^{i+43}(E(k)-E(i))^2$$

Puis, en effectuant une régression linéaire, on obtient une équation de C :

$$C = -0.0025714 \times V + 1.5142857$$

En analysant le morceau de musique précédent avec l'algorithme fraîchement modifié, on obtient les résultats de la figure 4. La figure supérieure montre la forme du signal, ainsi que les pulsations détectées, grâce à la constante C = 1.3 (traits supérieurs bleus) et grâce à C déterminé par la variance (traits inférieurs rouges). Les deux figures suivantes montrent la courbe de la différence avec C = 1.3 (bleue) et C déterminé par la variance (rouge).

Malgré une amélioration notable des résultats, ce n'est pas encore satisfaisant : il y a 5 fausses détections. Après de nombreux essais sur des morceaux de divers genres, il s'avère que cet algorithme produit de piètres résultats. On peut les expliquer par le fait que toutes les fréquences du signal sont mélangées lors du calcul des énergies. En effet, lors de l'analyse d'un signal comportant deux instrument, une flûte et une guitare par exemple, qui produisent à tour de rôle une note d'amplitude constante, l'algorithme ne verra aucune variation d'énergie et donc aucune pulsation !

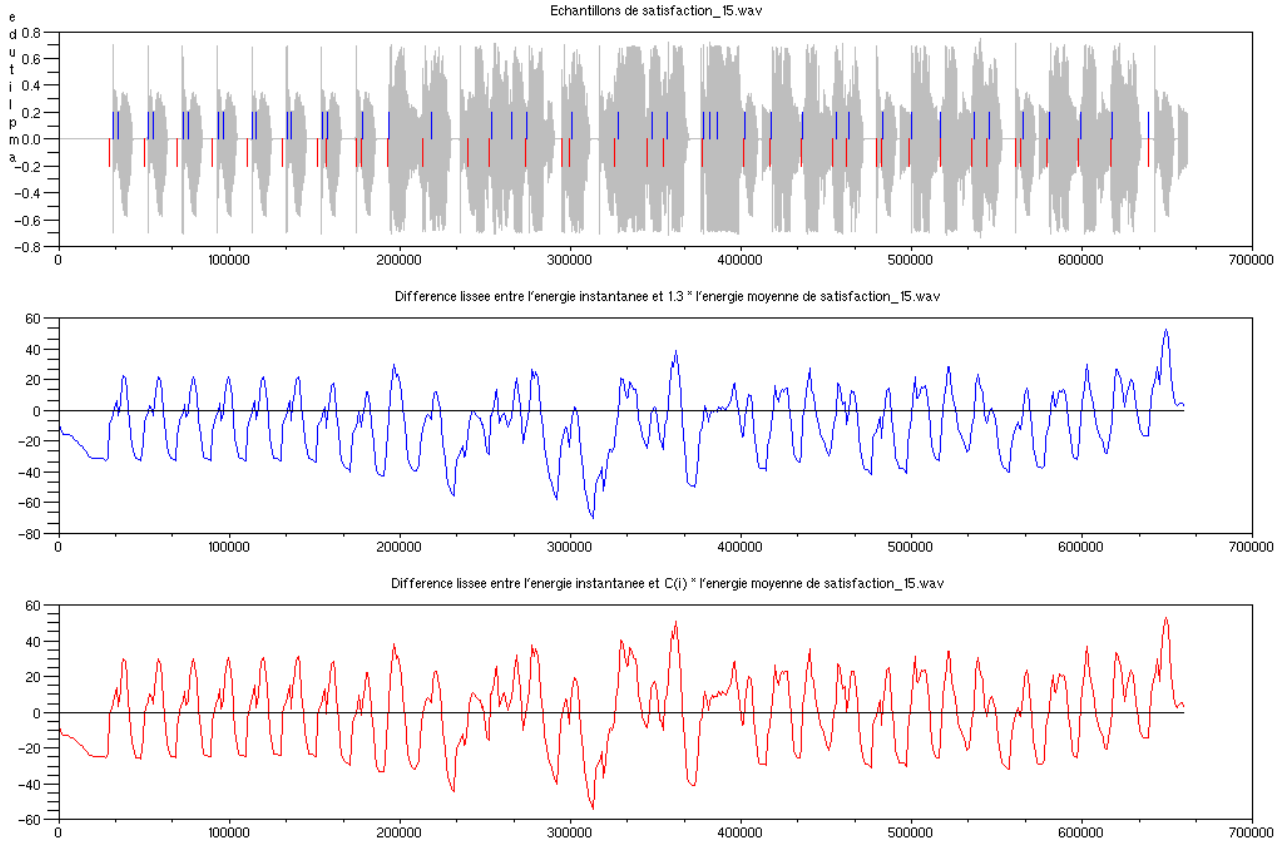

Illustration 4: Détermination de C par la variance

Une solution à ce problème est d'analyser les énergies dans des bandes de fréquences séparées. Pour cela, on passe dans le domaine fréquentiel grâce à la transformée de Fourier, et on calcule l'énergie contenue dans chaque bande de fréquences (théorème de Parseval). Un problème se pose cependant, lorsqu'il s'agit de collecter les résultats des différentes bandes de fréquences : dans combien de bandes doit apparaître une pulsation pour être prise en compte ? Comment traiter les pulsations légèrement décalées entre deux bandes de fréquences ?

À ces questions, il n'existe pas de réponse triviale, c'est pourquoi nous avons préféré utiliser un algorithme plus général à base de **flux d'énergie spectrale.**

3.2 - Flux d'énergie spectrale

Cet algorithme est basé sur un travail de recherche de M. Alonso [1]. Il est composé de trois parties : une fonction de détection des attaques (onsets), d'estimation de la périodicité et de localisation des pulsations. Dans notre cas, nous nous sommes intéressé uniquement à la première fonction.

Le but de cette fonction est de détecter les fonctionnalités les plus saillantes d'un signal, telles que les changements de notes et les percussions. Ces événement sont particulièrement important dans la détection d'attaques. Les attaques sont en générales masquées par des sons d'amplitude constante et plus élevée, c'est pourquoi une fonction de détection dans le domaine temporel ne les détecte pas.

La fonction est définie dans [1] de la manière suivante :

Le signal d'entrée, un vecteur x, est analysé en utilisant une transformée de Fourier fenêtrée (STFT). C'est à dire que de courtes portions de signal sont extraites à intervalles réguliers, multipliées par une fenêtre d'analyse (ici de Hanning) et transformées dans le domaine fréquentiel par le biais d'une transformée de Fourier :

$$\tilde{X}(f,m)=\sum_{n=0}^{N+1} w(n)\,x(n+mM)\,e^{-2j\pi f}$$

où x(n) est le signal d'entrée, w(n) est la fenêtre d'analyse de taille N, M est le pas de la STFT, m l'index de la portion de signal analysée et f la fréquence.

Le flux d'énergie spectrale est défini par J. Laroche [2] comme la dérivée du contenu fréquentiel d'un signal par rapport au temps :

$$E(f,k)=\sum_{m} h(m-k)\,G(f,m)$$

où h(m) approxime un filtre dérivateur avec :

$$h(e^{j2\pi f})\simeq j2\pi f$$

et la transformation

$$G(f,m)=F(\left|\tilde{X}(f,m)\right|)$$

est obtenue en deux temps : un filtrage passe-bas et une compression non linéaire. Nous avons suivi les conseils prodigués par M. Alonso pour la compression non linéaire, nous utilisons la fonction logarithme. Le filtrage passe-bas peut-être réalisé par un dérivateur du premier ordre `[1, -1]` ou bien par un filtre à réponse inpulsionnelle finie (FIR). Nous n'avons pas noté de différence notable qui ferait préférer l'un de ces deux filtres.

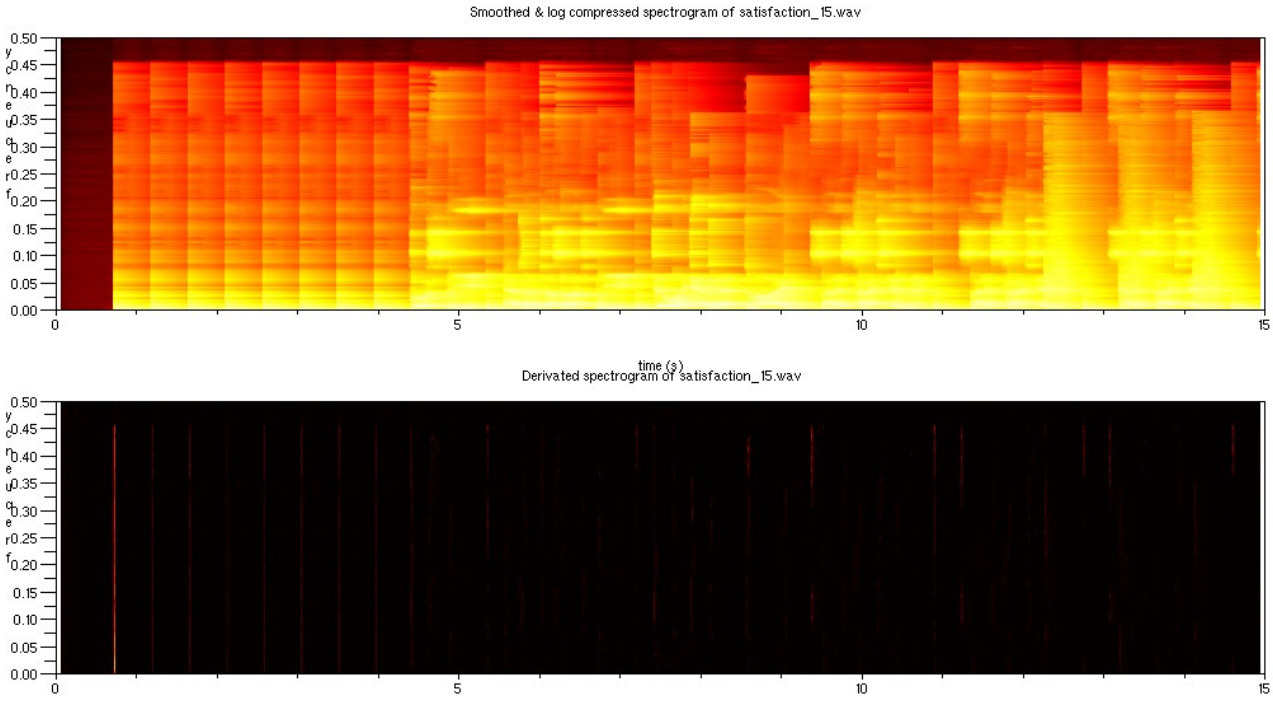

Illustration 5: STFT & SEF des 15 premières secondes de "Satisfaction" de Benny Benassi.

La figure 5 montre la STFT des 15 premières secondes de "Satisfaction" de Benny Benassi. On notera l'aspect « strié » du graphe, chaque strie représentant une attaque qui se reporte sur le flux d'énergie spectrale par une raie.

La figure 6 montre le signal d'entrée et les pulsations détectées.

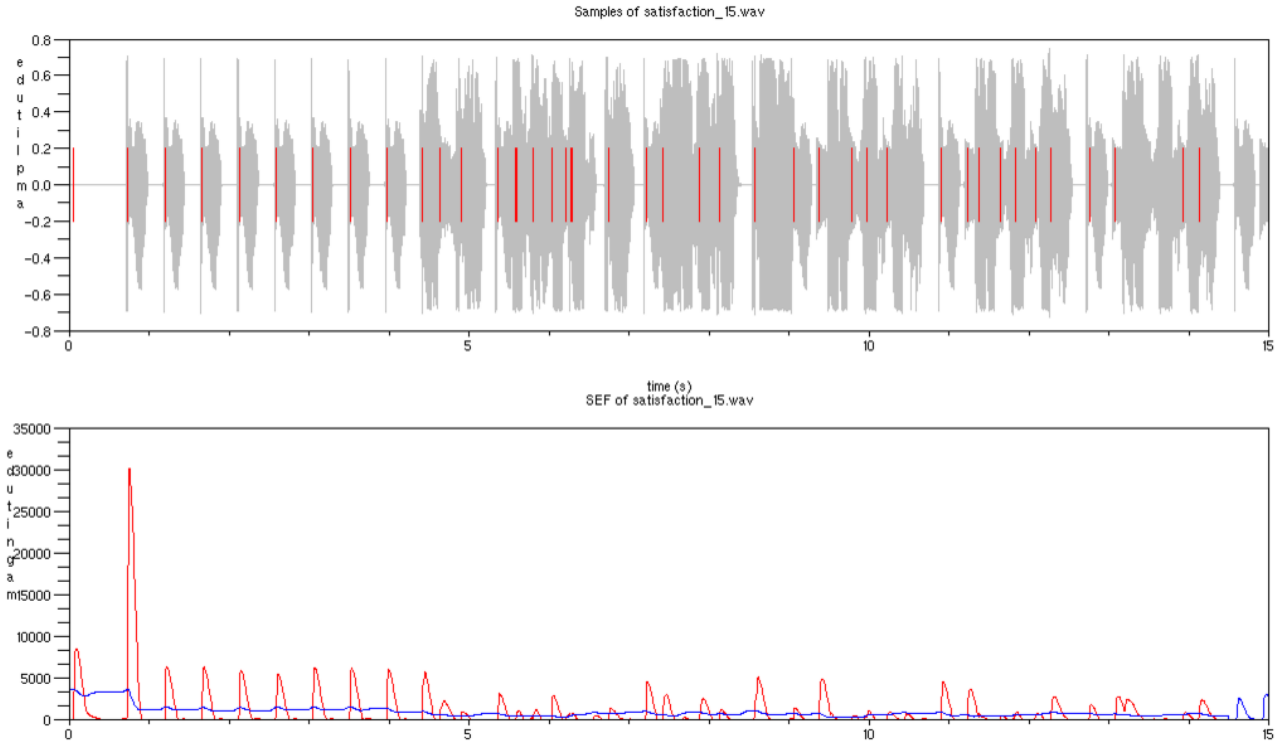

Illustration 6: Signal d'entrée et pulsations détectées

On remarque que cet algorithme ne détecte pas toutes les pulsations (faux négatifs), et induit de fausse détections (faux positif). Il est tout de même intéressant de noter qu'il produit de bons résultats dans de nombreux cas.

3.3 - Estimation du rythme à partir des dates des attaques

Nous avons concentré nos recherches autour de la détection des attaque, partie requise pour toute analyse qui pourrait suivre, et n'avons donc que très peu étudié la manière de traiter ces informations. Nos lecture ont tout de même révélé que la méthode la plus fréquemment utilisée pour calculer le tempo est l'utilisation de la résonance avec un peigne d'impulsion dont la fréquence est connue. Le calcul de la résonance est établi en calculant l'énergie du signal résultant de la corrélation du signal présentant des pics aux instants des attaques détectées avec le peigne d'impulsions, ceci pour plusieurs peigne de fréquence croissante. Le tempo correspond à la fréquence du peigne pour laquelle l'énergie calculée est maximum. Comme on peut s'y attendre, ce calcul est particulièrement coûteux, surtout que pour améliorer la précision du résultat on doit augmenter le nombre de peigne à tester. Cette méthode est donc difficilement compatible avec un traitement en temps réel.

Il est possible d'utiliser des méthode statistique pour arriver à un résultat approchant, mais nous ne les avons pas étudié, faute d'avoir de temps.

4 - Conclusion

L'avancement du projet à la date de rendu du rapport n'atteins pas les objectifs qui ont été fixés. En effet amaroK ne dispose toujours pas d'un module de détection du rythme, pas plus que nous ne disposons d'une implémentation efficace d'un algorithme de détection du rythme.

Cependant, nous pouvons d'ores et déjà tirer profit du travail réalisé. Tout d'abord le sujet nous a permis de voir sous un jour nouveau les enseignements de première année concernant la représentation des signaux et des systèmes ainsi que le traitement du signal. Les notions qui y ont été abordées font maintenant réellement partie des outils qui sont à notre disposition. En effet la conduite de ce projet a permis a chacun de revoir, d'appliquer et de mieux comprendre les points traités durant ces cours. Par ailleurs cela nous a permis de développer des compétences non négligeables dans l'écriture des scripts Scilab, compétence qui nous permettra à l'avenir de tester rapidement nos algorithmes sans avoir besoin pour cela tout écrire dans un langage généraliste tel que C.

Bibliographie

1: M. Alonso, B. David, and G. Richard, Tempo and beat estimation of musical signals, 2004

2: J. Laroche, Efficient Tempo and Beat Tracking in Audio Recordings, 2003