Cette année pour mon TIPE j’ai décidé d’étudier la reconnaissance musicale. Je vais donc vous présenter quelques généralités sur cette dernière. Je vais essayer de vous expliquer l’algorithme centrale de la reconnaissance musicale : l’algorithme d’empreinte. Finalement je vais conclure sur les résultats que j’ai obtenus avec cette algorithme.

Tout d’abord le principe de la reconnaissance musicale. Le but ultime d’un tel programme et sde reconnaitre peu importe le morceau et sa durée, son origine. Bref associé à un morceau une donnée connue.

L’algorithme repose sur un principe simple, on trace le spectrogramme du signal. Pour cela il faut tout d’abord transformer le signal stéréo en mono, c’est-à-dire passer d’une signale à deux voies (droite gauche) à un signale à une voie. Pour ma part je n’ai travaillé qu’avec des fichier wav codé sur 24 bit. D’ailleurs à l’aide de Will Drevo diplômé du MIT, j’ai créé programme permettant de lire ces fichiers.

Ensuite le but est de déterminer les amplitudes les plus fortes celle qui caractérise le morceau, et à partir de cela créer une empreinte qui sera presque unique. En effet l’empreinte de ce morceau ne sera associée qu’à l’empreinte de la musique dont le morceau est extrait

Ce schéma explique en partie le fonctionnement réel d’un tel programme. En effet pour chaque nouvelle musique le programme va incrémenter la base de donnée avec une nouvelle empreinte. Côté utilisateur, le programme doit faire l’empreinte de chaque morceau qui lui est soumis et le comparer avec toutes les empreintes de la base de donnée jusqu’à trouver celle qui convient. Le côté serveur est pour ma part mon ordinateur.

Maintenant il faut comprendre cet algorithme d’empreinte qui est centrale. Le but premier est de trouver les maxima d’amplitude. Pour cela j’ai utilisé une structure binaire comme il suit. La FFT, ou Fast fourier transform, renvoit un array, un tableau dont l’indice des colonnes représente le temps et celle des lignes les fréquences et le contenu un équivalent de l’amplitude associé à un temps et une fréquence. Le but est de caractériser fréquences pour lesquelles on obtient un maximum local. C’est ici qu’intervient la structure binaire ci-contre. Elle permet de discrétiser les valeurs du tableau obtenu par FFT. Ici c’est une structure itérée 5 fois et en réalité dans mon programme je l’ai itéré 20 fois.

Le principe est simple on utilise un filtre correspondant à cette structure sur un maximum et dans un nouveau tableau autour de chaque maximum dans un rayon de 20 cases on aura la même amplitude. Dans un cas idéal on obtiendrai un tableau d’amplitude bien discrétiser tel qu’il est ici. Mais on observe qu’en réalité cela va être rarement le cas à cause des frontières comme dans cet exemple. Cependant cela ne va pas gêner car le but est d’obtenir un tableau d’amplitude discrète.

En effet il suffit de comparer le tableau discrétiser et le tableau réel pour obtenir un couple temps fréquences correspondant à un maximum d’amplitude. On récupère ainsi une liste de triplet contenant une amplitude maximale local associé à un temps et une fréquence.

On a donc réussi à localiser les amplitudes maximales les coordonées etant le temps et la fréquence associé. Maintenant vient la deuxième partie de l’algorithme d’empreinte comment encoder une empreinte de telle sorte qu’elle soit unique. Tout d’abord qu’est ce qu’une empreinte. Pour ma part à l’origine c’est juste l’association de deux fréquences, on verra par la suite qu’il est plus judicieux d’introduire aussi une différence de temps. Mais evidement un morceau de 3 minutes contient plus d’1 milliers d’amplitudes maximales locale. L’enjeu de cette algorithme est de déterminer une valeure à partir de laquelle 2 fréquences sont trop éloigné pour être appariés dans une empreinte. Par exmple j’ai choisi une valeur de 20 unité de temps pour cette algorithme. Si 2 fréquences sont éloginés de plus de 20 unités de temps elles ne seront pas appariés. C’est l’enjeu de l’algorithme car plus le degré est faible moins il y aura d’empreinte mais cela implique aussi une plus grande difficulté de reconnaissance. A l’inverse plus le degré est fort plus il y aura d’empreinte, plus le morceau sera unique, mais le problème devient alors le stockage de ces empreintes.

Alors pour encoder une empreinte j’utilise une structure de générateur et les fonctions de hachage . Ces fonctions permettent à une chaîne de caractère (ici de quoi est formé l’empreinte) d’être codé sous formes de chiffres et de lettre c’est une simple fonction de cryptage comme on peut l’apercevoir au tableau. Et donc si c’est interessant c’est que pour réduire la taille des empreintes on peut supprimer les premiers bits d’une empreinte après être passé dans la focntion de hachage. Cela réduit l’unicité de l’empreinte, donc il y a une nouvelle fois un degrés à choisir.

Maintenant il faut comprendre en quoi cela nous permet de reconnaitre une musique. La première approche, la plus simple que j’ai eu est de comparer uniquement des morceaux ayant la même durée. On compare au cas par cas le nombre d’empreinte correspondant et on renvoit la musique ayant le pourcentage de correspondance le plus grand. Le problème ici est que l’offset de temps doit être le même pour avoir une correspondance. Cependant ce programme marche très biens même avec saturations du signal.

La 2ème approche est d’introduire la différence de temps entre les deux fréquences. Cette fois ci l’empreinte n’est plus repérée par un offset de temps qui est propre a l’extrait mais par une différence de temps qui elle est propre au morceau entier. La comparaison reste la même. On compare les empreintes de l’extrait avec chaque empreinte de la base de données et on renvoit la muisque avec le meilleur pourcentage.

Le fonctionement de la base de donnée. On enregistre les empreintes des musiques à reconnaitre et on incrémente la base de donnée avec les métadonnée de la musique.

Passons maintenant aux résultats ici ce sont les spectrogrammes avec les maxima d’amplitudes localisées par des points