

Restoration d'enregistrements sonores

Niveau: facile

Stéphane Lengrand
lengrand@lix.polytechnique.fr

26th January 2010

L'objectif de ce projet est de restorer des enregistrements sonores digitaux qui ont été abîmés. Rappelons qu'un enregistrement digital est basé sur une discrétisation du signal qui devient alors *numérisé* : la valeur de celui-ci est enregistrée (par exemple) 44100 fois par seconde sous forme d'une succession d'*échantillons*. Nous nous intéressons à un problème particulier, celui de l'écrtage. Ce problème peut être d'origine digitale ou analogique.

Un écrtage digital peut avoir été causé par une opération sur le signal numérisé pour augmenter son volume : typiquement, si les échantillons son chacun enregistrés sur 1 octet, ils ont dépassé la valeur 127 (ou -128).

Un écrtage analogique peut être dû aux limites physiques des appareils utilisés avant la numérisation, comme par exemple un micro : si la source sonore est trop forte / trop près du micro, il est possible que sa membrane ne puisse pas s'écarter suffisamment de sa position au repos pour convertir fidèlement les vibrations en signal électrique.

Le projet comprend trois tâches :

- Choisir un format de données audio (par exemple le format `wav`) et manipuler ses contenus (lecture/écriture) au travers de classes java
- Repérer les sections du signal qui ont été écrtés
- Interpoler ce que pouvait être le signal d'origine dans ces sections.

Pour la première tâche, le format de données audio le plus simple à manipuler est vraisemblablement le format `wav` : il n'est pas compressé, si bien que les échantillons du signal sont enregistrés les uns à la suite des autres sur les octets du fichier. Typiquement, 1 échantillon = 1 octet (donc variant entre -128 et 127), mais il peut y avoir des variantes : chaque échantillon peut être enregistré sur 2 octets, ou alors si le signal est stéréo le fichier alterne entre un échantillon du canal gauche et un échantillon du canal droit, etc. Un petit en-tête, au début du fichier, contient ce genre de paramètres ainsi que la fréquence d'échantillonnage (44100 Hertz est celle des CD traditionnels, par exemple). Il est possible que java fournisse déjà des outils (des classes) pour lire et écrire des fichiers `wav` sans que vous deviez vous préoccuper du format de fichier ; dans le cas contraire, la description ci-dessus¹ devrait vous permettre d'écrire, sans trop de difficultés, vos propres méthodes de lecture et d'écriture, afin de manipuler vos signaux sonores par les structures de données qui vous paraissent les plus adéquates pour les tâches suivantes (par exemple, un tableau d'entiers).

¹complétée par une petite recherche sur le web pour comprendre le format de l'en-tête

La deuxième tâche est clairement plus facile lorsque l'écrêtage est d'origine digitale, car les "crêtes" sont très propres, elles sont des parties du signal exactement constantes (de dérivée nulle) alors que leurs débuts et leurs fins sont des points de discontinuité de la dérivée. Lorsque l'écrêtage est d'origine analogique, les caractéristiques des crêtes sont les mêmes mais un peu plus floues, car l'atteinte des limites physiques des appareils peut se traduire de manière plus subtile dans le signal enregistré. Des tests seront utiles pour calibrer les critères.

Dans tous les cas il sera ainsi utile de programmer le calcul de la dérivée première et seconde, pour repérer parties constantes et discontinuités. La plupart des outils de traitement de fichier `wav` permettent, si l'on zoome suffisamment, de visualiser les échantillons du signal (et, pourquoi pas, des dérivées).

La troisième tâche consiste, pour chaque section identifiée comme écrêtée, à jeter les échantillons présents et à les remplacer par une interpolation du signal original.

Soit s la fonction signal et \mathcal{I} l'interpolant. La section de l'interpolant qui remplace la zone où s est à reconstituer doit se raccorder au reste du signal, et ceci aux 2 points que sont le début et la fin de la section. Si l'on appelle x_1 et x_2 leurs abscisses et $s(x_1)$ et $s(x_2)$ leurs ordonnées, on veut

$$\begin{aligned}\mathcal{I}(x_1) &= s(x_1) \\ \mathcal{I}(x_2) &= s(x_2)\end{aligned}$$

On peut aller plus loin en demandant l'égalité des dérivées

$$\begin{aligned}\mathcal{I}'(x_1) &= s'(x_1) \\ \mathcal{I}'(x_2) &= s'(x_2)\end{aligned}$$

et plus généralement

$$\begin{aligned}\mathcal{I}^{(i)}(x_1) &= s^{(i)}(x_1) \\ \mathcal{I}^{(i)}(x_2) &= s^{(i)}(x_2)\end{aligned}$$

La piste la plus simple pour un interpolant, et celle que développera en premier lieu ce projet, sera d'utiliser un polynôme pour \mathcal{I} . Ses coefficients seront trouvés en résolvant les équations ci-dessus. Un polynôme de degré $2i + 1$ ($2i + 2$ coefficients à trouver) vous permettra de satisfaire l'égalité des i premières dérivées, en résolvant par un pivot de Gauss un système linéaire de $2i + 2$ équations à $2i + 2$ inconnues :

- un polynôme de degré 1 vous permettra d'assurer la continuité avec le reste du signal par une ligne droite (autrement dit, une crête parfaite), ce n'est donc pas ce que l'on veut,
- un polynôme de degré 3 vous permettra d'assurer la continuité du signal et de sa dérivée première
- etc

En pratique, nous commencerons par regarder les résultats que donnent les polynômes de degré 3.

En effet il n'y a pas grand intérêt à assurer la continuité de nombreuses dérivées car nous n'avons de toute façon qu'un échantillonnage discret du signal. Par exemple, les valeurs $s^{(i)}(x_1)$ et $s^{(i)}(x_2)$ seront approximées grâce aux i échantillons juste avant x_1 et juste après x_2 , donc plus i est grand plus l'approximation est grossière.

On pourra ensuite regarder comme interpolant des fonctions sinusoidales.