Le code correcteur d'erreurs du CD audio

Sujet proposé par Nicolas Sendrier

`Nicolas.sendrier@inria.fr`

Important, ce projet est accompagné d'une *page de suivi*[1].

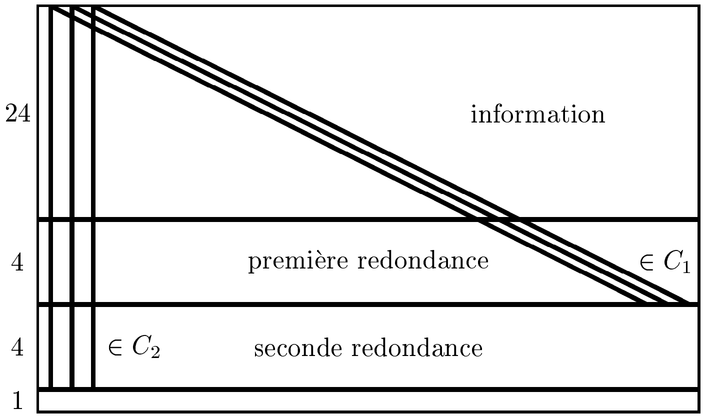
1 Présentation générale

Le standard du disque compact audio comporte un code correcteur d'erreurs appelé CIRC (*Cross Interleaved Reed-Solomon Code*). Ce code a pour but de lutter contre les erreurs de lecture. Ces erreurs peuvent provenir d'imprécisions mécaniques, de défauts microscopiques du support, de traces de doigts, de rayures ...

Pour combattre ces dégradations de l'information, un octet sur quatre inscrits sur le CD est redondant et sert à corriger ces erreurs.

2 Le code CIRC

Le schéma de codage du CD audio utilise deux codes de Reed-Solomon, $C_1 = RS(28, 24)$ et $C_2 = RS(32, 28)$ sur $\mathbf{F}_{256}$ (le corps à 256 éléments). L'écriture $RS(n, k)$ est utilisée pour désigner un sous-espace vectoriel de dimension $k \leq n$ de l'espace vectoriel $\mathbf{F}_{256}^n$ (cf. l'appendice A).



L'information musicale est découpée en bloc de 24 octets et placée dans une table d'entrelacement de 24 lignes. La redondance permettant la correction des erreurs est ajoutée en deux fois, la première fois en diagonale à l'aide du code C_1, la seconde fois verticalement à l'aide du code C_2. Un trente-troisième octet de service est ajouté au bout de chaque colonne.

[1] `http://www.enseignement.polytechnique.fr/profs/informatique/Nicolas.Sendrier/X02/IF/projet.html`

2.1 Codage

Les procédures de codage pour C_1 et C_2 se décrivent mathématiquement de la façon suivante :

$$\begin{array}{rcl} \mathbf{F}_{256}^{k} & \to & \mathbf{F}_{256}^{k+4} \\ (x_1, \ldots, x_k) & \mapsto & (x_1, \ldots, x_k, x_{k+1}, x_{k+2}, x_{k+3}, x_{k+4}) \end{array}$$

avec

$$(x_{k+1}, x_{k+2}, x_{k+3}, x_{k+4}) = (x_1, \ldots, x_k)R$$

où k vaudra 24 ou 28 respectivement pour les codes C_1 et C_2. Dans les deux cas les quatres symboles supplémentaires dépendent linéairement des k premiers. La matrice R est de taille $k \times 4$ (et sera différente pour C_1 et C_2).

Le schéma de codage complet est le suivant :

- l'information est découpée en blocs de 24 octets,
- chaque bloc est codé en un mot de C_1,
- les mots de C_1 sont placés dans une table d'entrelacement à retard 4 (cf. appendice B),
- chaque colonne de la table est codée en un mot de C_2,
- les données sont inscrites sur le disque colonne par colonne.

Le détail des procédures de codage de C_1 et C_2 n'est pas utile à ce stade, elles seront explicitées sur la page web du projet.

2.2 Décodage

Les deux codes considérés ont une distance minimale 5, c'est-à-dire qu'ils permettent de corriger jusqu'à 2 erreurs ou 4 effacements (cf. l'appendice A).

La procédure de décodage est appliquée à la table d'entrelacement. Nous avons successivement :

1. Un décodeur d'une erreur pour C_2 qui agit de la façon suivante sur chaque colonne $y \in \mathbf{F}_{256}^{32}$ de la table d'entrelacement :
   - s'il existe un mot x dans C_1 tel que $d_H(x, y) \leq 1$ alors l'éventuel symbole erroné est modifié,
   - sinon tous les symboles de la colonne sont marqués comme effacés.
2. Un décodeur de quatre effacements pour C_1 qui agit de la façon suivante sur toute diagonale $y \in \mathbf{F}_{256}^{28}$ de la table d'entrelacement (tronquée aux 28 premières lignes) :
   - si moins de quatre symboles sont effacés, on cherche le seul mot de C_1 coincidant avec les symboles non effacés ; s'il existe, on remplace les symboles effacés par les valeurs de ce mot,
   - si plus de quatre symboles sont effacés, ou si l'on n'a pas trouvé de mot de C_1, tous les symboles de la diagonale sont marqués comme effacés.

Le détail des procédures de décodage de C_1 et C_2 n'est pas utile à ce stade, elles seront explicitées sur la page web du projet.

3 Travail demandé

Il s'agit de programmer un démonstrateur qui incluera :

- le codage,
- une modélisation des erreurs,
- le décodage.

Pour mettre en place la démonstration, l'utilisation de fichiers audio n'est pas pertinente car il est difficile d'entendre les erreurs. En revanche avec des fichiers d'image (typiquement au format ppm), il est possible de percevoir les erreurs. Par exemple dans l'exemple ci-dessous on voit l'effet d'une rayure (en bas à droite) sur un fichier codé.

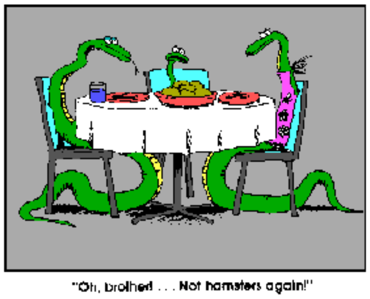
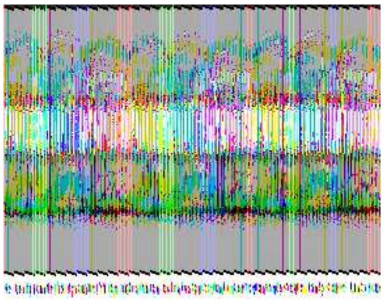
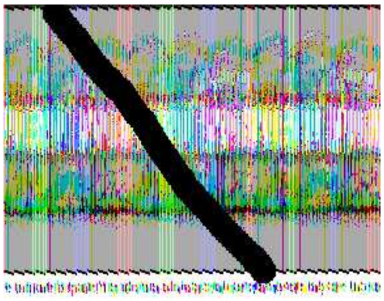


Image originale



Information après entrelacement



Une rayure

Effet de la rayure sur l'information

3.1 Codage et décodage

À l'entête des fichiers près (qu'il faudra parser et adapter) il s'agit du codage et du décodage décrit précédemment. Un fichier codé sera plus gros que le fichier original. On pourra le manipuler comme une image en rallongeant les lignes, par exemple :

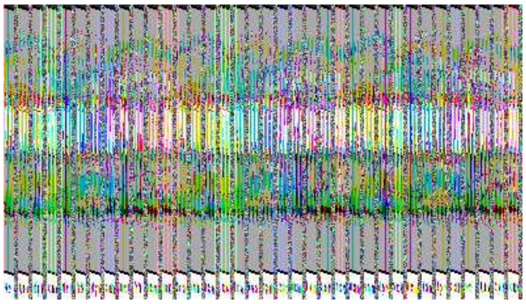


Image entrelacée codée

3.2 Modélisation des erreurs

Les erreurs se produiront dans le fichier codé et sont de deux types :

- des erreurs isolées, qui affectent un seul octet et qui correspondent à des imprécisions de lecture ou à des défaut sur le plastique. Le taux de ces erreurs est très faible en général (moins d'un octet sur mille), mais il pourra être intéressant de voir ce qui se passe lorsqu'il augmente.
- des erreurs en rafales, qui affectent un grand nombre d'octets consécutifs qui peu aller jusqu'à quelques centaines voire quelques milliers.

Il y a deux approches possibles pour ajouter des erreurs au fichier codé. La première consiste à écrire un programme spécialisé qui lira le fichier, et, selon un modèle aléatoire qu'il faudra définir précisement, le modifie. L'autre approche consiste à utiliser un logiciel de traitement d'image (gimp par exemple) pour ajouter une ou plusieurs rayures au fichier codé.

3.3 Première extension : interpolation

Il existe un second mécanisme de protection de l'information dans le standard du CD audio, qui consiste à espacer au maximum les échantillons musicaux (pour nous les pixels) consécutifs de façon à pouvoir interpoler l'information manquante si un mot de code venait à ne pas être décodé. Des informations complémentaires seront fournies sur la page web.

3.4 Seconde extension : Analyse de la capacité de correction

Il s'agit d'une extension à géométrie variable. Une fois le démonstrateur achevé, il peut être utilisé pour effectuer des simulation d'erreurs et de correction pour divers modèle d'erreurs significatifs. Il s'agit de faire varier le taux des erreurs isolées, le nombre et la longueur des rafales, et d'étudier la qualité de l'information corrigée en fonction de tous ces paramètres.

A Distance de Hamming et codes correcteurs

Soit $\mathbf{F}$ un corps fini et n un entier positif. La *distance de Hamming* sur $\mathbf{F}^n$ est définie entre deux mots $x = (x_1, \ldots, x_n)$ et $y = (y_1, \ldots, y_n)$ de $\mathbf{F}^n$ par :

$$d_H(x,y) = |\{i \mid 1 \leq i \leq n, x_i \neq y_i\}|.$$

C'est-à-dire qu'il s'agit du nombre de coordonnées différentes.

Un code correcteur d'erreur linéaire de longueur n sur $\mathbf{F}$ est un sous-espace métrique non vide de $\mathbf{F}^n$ muni de la distance de Hamming. La dimension d'un code linéaire est sa dimension en tant que sous-espace vectoriel de $\mathbf{F}^n$, elle est traditionnellement notée k. Nous parlerons de code (n,k), bien sûr $k \leq n$.

La *distance minimale* d'un code est la plus petite distance entre deux mots distincts du code. Il s'agit d'un paramètre important puisque qu'elle fixe la capacité de correction d'un code.

Proposition 1 *Un code correcteur de distance minimale d peut corriger jusqu'à $(d-1)/2$ erreurs.*

Autrement dit si x est un mot de code et si y vérifie $d_H(x,y) \leq (d-1)/2$, alors il n'existe pas d'autre mot de code plus proche de y pour la distance de Hamming.

Correction d'effacement. Lorsqu'un symbole dans un mot est erroné mais que sa position est connue, on parle d'effacement. Il s'agit d'une situation plus favorable en terme de capacité de correction.

Proposition 2 *Un code correcteur de distance minimale d peut corriger jusqu'à $d-1$ effacements.*

En effet, si $d-1$ positions d'un mot de code x sont effacées, il n'existe aucun autre mot de code qui coincide sur les $n-d+1$ positions restantes.

B Entrelacement à retard

L'entrelacement est une technique classique en télécommunication consistant à mélanger l'information avant de la transmettre. Dans le standard du disque compact il s'agit d'un entrelacement *à retard*. Par exemple, pour entrelacer 20 symboles, de x_1 à x_{20}, avec une profondeur 4 (taille des blocs) et un retard 1, on remplit une table d'entrelacement en diagonale de la façon suivante :

x_1	x_5	x_9	x_{13}	x_{17}	$\bullet$	$\bullet$	$\bullet$
$\bullet$	x_2	x_6	x_{10}	x_{14}	x_{18}	$\bullet$	$\bullet$
$\bullet$	$\bullet$	x_3	x_7	x_{11}	x_{15}	x_{19}	$\bullet$
$\bullet$	$\bullet$	$\bullet$	x_4	x_8	x_{12}	x_{16}	x_{20}

L'ordre après entrelacement sera :

$$x_1,\ \bullet,\ \bullet,\ \bullet,\ x_5,\ x_2,\ \bullet,\ \bullet, x_9,\ x_6,\ x_3,\ \bullet,\ x_{13},\ x_{10},\ x_7,\ x_4,\ x_{17}, x_{14},\ x_{11},\ x_8,$$
$$\bullet,\ x_{18},\ x_{15},\ x_{12},\ \bullet, \bullet,\ x_{19},\ x_{16},\ \bullet,\ \bullet,\ \bullet,\ x_{20}$$

Remarques :

1. Il faut remplir avec des valeurs arbitraires ($\bullet$) le début et la fin de la table.
2. La table peut contenir un nombre d'élément égal à n'importe quel multiple de la profondeur de l'entrelacement. Ce multiple n'a pas besoin d'être connu à l'avance.

On peut augmenter le retard en changeant la pente de la diagonale. Par exemple un retard 2 signifie que le début de la table serait :

x_1	x_5	x_9	x_{13}	x_{17}	$\cdots$
$\bullet$	$\bullet$	x_2	x_6	x_{10}	$\cdots$
$\bullet$	$\bullet$	$\bullet$	$\bullet$	x_3	$\cdots$
$\bullet$	$\bullet$	$\bullet$	$\bullet$	$\bullet$	$\cdots$

La table d'entrelacement du disque compact est à retard 4 (x_2 est sur la deuxième ligne de la cinquième colonne).