Génie Mathématique et Modélisation
Quatrième Année

T.P. COMPRESSION D'IMAGES PAR CODAGE EN SOUS-BANDES

Philippe GUILLAUME

Jean-Paul VILA

2014

1 Séance d'introduction

L'objet de la première séance est de faire quelques expériences de filtrage d'images.

Vous trouverez à l'adresse `www-gmm.insa-toulouse.fr/~guillaum/tpsignal` trois fichiers `escher.mat`, `barbr.mat` et `lenar.mat` contenant des images, les deux dernières étant souvent utilisées en tests de compression d'image. D'autres fonctions que vous aurez à utiliser au fur et à mesure se trouvent à cette adresse. Le fichier de couleurs `map.mat` (niveaux de gris) sera utilisé par la fonction `colormap` de Matlab. Copiez ces fichiers dans votre répertoire, puis, pour visualiser une image sous Matlab, lancez la commande

```
load nom_du_fichier; load map;
```

et utilisez ensuite les fonctions Matlab `image` et `colormap.` Les images ont pour nom de variable `IM`. N'hésitez pas à utiliser la commande `help`.

Pour les opérations de filtrage qui suivent, vous pourrez considérer l'image comme un signal mono-dimensionnel. Pour cela, mettez bout à bout les lignes de la matrice de l'image via la commande Matlab `reshape`. Le filtrage se fait avec la commande

```
y = filter(b,a,x).
```

C'est à vous de fournir la transformée en z, $H(z) = b(z^{-1})/a(z^{-1})$, les vecteurs `a` et `b` étant formés des coefficients des polynômes en z^{-1}, a et b. Les images étant stokées sous format entier `uint8`, le filtrage devra être précédé d'une conversion de type : `x = double(x)`.

Remarque 1. Attention, le résultat `y` fourni par la fonction Matlab `filter` a la même longueur que le signal d'entrée `x`. Dans le cas de filtres RIF, si vous souhaitez un filtrage sans retard ou sans perte d'information, vous pouvez au préalable rajouter en fin du signal `x` le nombre de zéros nécessaires et tronquer ensuite le début du signal de sortie. Une autre possibilité consiste à prolonger le signal `x` par périodicité ou par symétrie. La fonction `filtre.m` (toujours sous `/tpsignal`) effectue un prolongement par symétrie, et s'applique au cas de filtres symétriques de longueur impaire. Elle est utilisée par les fonctions `analyse` et `synthèse` de la troisième séance, et nécessite une décimation-interpolation portant sur les indices pairs ou impairs selon qu'il s'agit du canal passe-bas ou passe-haut. Cf. [1] pour plus de détails.

En utilisant Matlab, vous observerez l'effet sur une image des opérations suivantes :

1. filtrage passe-bas à phase linéaire : construire le filtre avec la fonction `fir1(n,Wn)` pour différentes valeurs de `n` (degré du filtre) et de `Wn` (fréquence de coupure),
2. filtrage passe-bas à phase non linéaire : filtre construit avec la fonction `butter(n,Wn)`,
3. filtrage passe-haut : filtre construit avec la fonction `fir1(n,Wn,'high')`,
4. filtrage passe-haut puis accentuation du contraste (mise à 0 ou 255 de chaque pixel),
5. décimation-interpolation (une valeur sur deux remplacée par zéro) puis filtrage passe-bas par un filtre à phase linéaire.

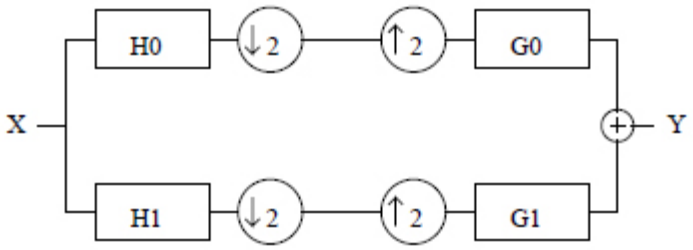
Vous utiliserez `freqz` et `freqzplot` pour voir les caractéristiques des filtres employés.

2 Deuxième séance : réalisation d'un banc de filtres à reconstruction parfaite

L'un des intérêts du codage en sous-bandes est la compression d'images. On rappelle d'abord le principe du codage en sous-bandes en dimension 1, puis on l'étend à la dimension 2. Cette séance est consacrée à la réalisation d'un banc de filtres à reconstruction parfaite. La prochaine séance sera consacrée à la compression proprement dite.

2.1 Codage en sous-bandes des signaux 1D

Le schéma du codage-décodage en deux sous-bandes d'un signal mono-dimensionnel est le suivant :



Le signal discret d'entrée x est filtré par deux filtres RIF, un filtre passe-bas h_0 et un filtre passe-haut h_1, appelé filtre complémentaire. Après décimation d'un facteur 2 (suppression d'une valeur sur deux, flèche vers le bas sur le schéma), on obtient deux signaux x_0 et x_1 (les deux sous-bandes). A la reconstruction, on interpole d'un facteur 2 (on intercale un zéro entre chaque valeur, flèche vers le haut sur le schéma), et on filtre les deux signaux obtenus avec les filtres RIF g_0 et g_1. En additionnant les deux signaux sortants, on obtient un signal y.

La transformée en z du signal reconstruit y s'écrit :

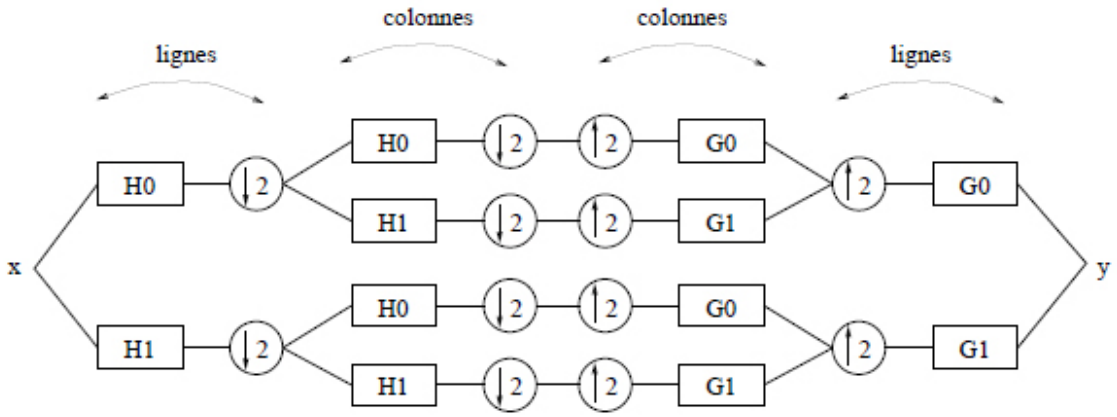
$$\begin{aligned} Y(z) &= \frac{1}{2}[H_0(z)G_0(z) + H_1(z)G_1(z)]X(z) \\ &+ \frac{1}{2}H_0(-z)G_0(z) + H_1(-z)G_1(z)]X(-z). \end{aligned}$$

Lorsque les quatre filtres sont choisis tels que l'on ait

$$\begin{aligned} &G_0(z) = H_1(-z) \\ &G_1(z) = -H_0(-z) \\ &H_0(z)G_0(z) + H_1(z)G_1(z) = C\,z^{-k}, \end{aligned}$$

le signal de sortie y est égal au signal d'entrée x, à un retard k et un facteur d'amplification $C/2$ près. C'est alors un banc de filtres à reconstruction parfaite.

2.2 Schéma du codage 2D séparable



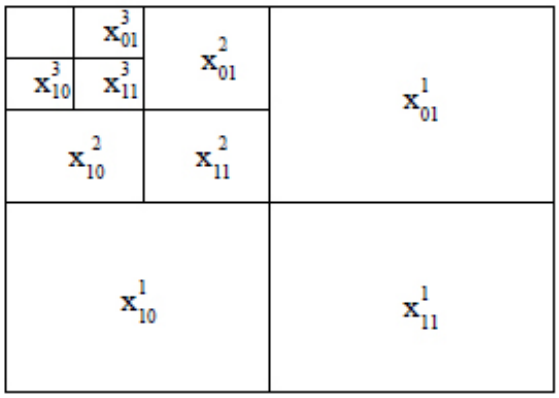
Pour les signaux 2D (images), on modifie le schéma précédent de la manière suivante : on filtre d'abord les lignes du signal d'entrée x (c'est une matrice) avec les filtres h_0 et h_1, ce qui donne deux signaux x_0 et x_1 dont on décime les lignes d'un facteur 2. On filtre ensuite les colonnes de chacun de ces deux signaux avec les mêmes filtres, ce qui donne quatre signaux dont on décime les colonnes d'un facteur 2. On a donc à ce stade quatre sous-images que nous noterons x_{00}, x_{01}, x_{10} et x_{11}. La reconstruction commence par une interpolation d'un facteur 2 sur les colonnes, suivie d'un filtrage des colonnes avec g_0 et g_1. On interpole ensuite les lignes d'un facteur 2, et on termine par un filtrage des lignes avec de nouveau g_0 et g_1. La somme des quatre images en sortie du banc redonne un signal y qui doit être égal au signal initial x, avec un retard égal à k. La valeur de k est celle qui apparaît dans $H_0(z)G_0(z) + H_1(z)G_1(z) = C\,z^{-k}$, et peut être obtenue via la commande Matlab `conv`. Il faudra donc ajouter k zéros en fin de lignes et en fin de colonnes avant de faire passer l'image dans le banc de filtres.

1. En utilisant la fonction `litfiltres.m` pour lire les filtres `h0`, `h1`, `g0` et `g1` (toujours sous le répertoire `/tpsignal`), construisez le banc de filtres, et vérifiez la reconstruction parfaite en comparant les images x et y. Pour les visualiser, utilisez les commandes `image` et `colormap` de Matlab.

2. Il est également intéressant d'observer les quatre sous-images x_{ij} obtenues. Pour cela juxtaposez-les dans une seule matrice et visualisez-la avec les commandes `image` et `colormap`. Trois des sous-images ont une moyenne à peu près nulle. Pour mieux les voir, vous pouvez effectuer sur chacune d'elle une transformation affine appropriée pour obtenir des valeurs comprises entre 0 et 255.

3 Troisième séance : compression.

Le procédé de décomposition en sous-images décrit précédemment peut être de nouveau appliqué à la sous-image passe-bas-passe-pas x_{00}, ce qui donne deux niveaux d'analyse. On peut ainsi poursuivre jusqu''a une profondeur de 3, 4, ..., niveaux d'analyse. Pour trois niveaux par exemple, on obtient des sous-images disposées de la manière suivante :



Le cycle complet compression-décompression à effectuer comporte les étapes suivantes :

1. analyse de l'image jusqu'à une profondeur donnée,
2. quantification du résultat (cf. ci-dessous), écriture dans un fichier sous format binaire (commande Matlab `save`), et compression du fichier par la commande Unix `gzip`,
3. décompression du fichier (`gzip -d nom_du_fichier`) et lecture du fichier sous Matlab (commande Matlab `load`),
4. déquantification de l'image,
5. reconstruction de l'image (synthèse), visualisation du résultat (commande Matlab `image`).

Les fonctions `analyse.m` et `synthese.m` qui vous sont fournies effectuent simultanément les opérations d'analyse-quantification et de déquantification-synthèse. Il existe des procédés de compression plus sophistiqués que celui employé ici, et qui utilisent une lecture en zigzag des pixels des sous-images.

On note $x^k_{ij},\ 0 \leq i,j \leq 1,\ 1 \leq k \leq 4$ les sous-images obtenues au niveau k (avec $x^1_{ij} = x_{ij}$). Une fois chaque analyse effectuée, on quantifie chaque pixel de chaque sous-image x^k_{ij}. Ceci est effectué par la procédure `quant.m` qui contient en particulier les instructions suivantes :

```
function [y,a,b] = quant(x,q,s)
% pas de quantification = q/log(variance)
[nl,nc] = size(x);
xl = reshape(x,1,nl*nc);
mn = mean(xl);
pas = q/max(log(cov(xl)),log(1.1));
a = 1/pas;
b = -mn/pas;
y = round(a*x + b);
```

où q est un paramètre de compression à régler pour chaque sous-image. Cette opération effectuée, on récupère une image y composée de la juxtaposition des sous-images, que l'on sauve dans un fichier avec la commande Matlab

```
save nom_du_fichier round(y);
```

Ce fichier est ensuite compressé avec la commande Unix `gzip`.

Le travail que vous avez à faire est le suivant.

1. En utilisant les fonctions `analyse.m` et `synthese.m,` effectuez une analyse à 4 niveaux suivie d'une synthèse, et vérifiez que pour $q = 1$ l'image reconstruite est indiscernable à l'oeil de l'image d'origine.
2. En choisissant vous-même les paramètres q pour chaque niveau d'analyse, essayez d'obtenir un taux de compression de l'ordre de $10 \rightarrow 1$ sans que l'image reconstruite soit trop altérée. Vous pourrez utiliser les deux critères d'erreur suivants :

   ```
   er1 = max(max(abs(y-x)));
   er2 = sum(sum(abs(y-x)))/(nblignes*nbcolonnes);
   ```

   La comparaison sur la taille du fichier sera faite avec les fichiers `lenar.mat.gz` ou `barbr.mat.gz,` que vous aurez obtenus en utilisant la commande `gzip`. Vous pouvez aussi faire des comparaisons avec le format JPEG, en sauvant les images avec la commande Matlab `imwrite(x,'*.jpg')`, qui donne une compression sans pertes.

Travail à rendre

Vous rendrez un court rapport (maximum 4 pages) contenant vos remarques sur les différents filtrages de la première séance, et les résultats de compression que vous avez obtenus aux séances suivantes.

References

[1] G Strang and T Nguyen, Wavelets and Filter Banks, Wellesley, Cambridge Press, 1996.