Compte-Rendu de TP Signal:
Compression d'images par codage en sous-bandes

Nicolas Bonneel, Jean-Gabriel Prince

4ème année GMM, Insa Toulouse - Décembre 2004

Introduction

Le but de ces TP est d'appliquer les connaissances acquises en traitement du signal à la compression d'images. L'image est une matrice d'indices se référant à une palette de niveaux de gris (on travaille ici avec le dessin d'Escher). Dans un premier temps on essayera de restituer l'image originale sans pertes en la filtrant à l'aide de differents filtres décrits plus loin. Puis, on essayera de diminuer la taille du fichier de l'image d'un facteur 10 sans trop altérer la qualité de l'image.

1 Etude de différents filtres :

Le filtrage est opéré sur un signal mono-dimensionnel que l'on obtiendra en concatenant les lignes de l'image en un seul vecteur.

Remarque :

La transformée en z d'un filtre est définie commme suit : $H(z) = \frac{B(z^{-1})}{A(z^{-1})} = \frac{b_0+b_1z^{-1}+...+b_rz^{-r}}{a_0+a_1z^{-1}+...+a_qz^{-q}}$

1.1 Filtres passe-bas

1.1.1 Phase linéaire : RIF

Pour avoir un filtre RIF, il faut que le polynome A soit de degré nul (vu en cours).

On a alors un filtre dont les coefficients suivent une fonction `sinc` symétrique. La commande `Matlab` qui donne les coefficients du polynome B de ce filtre est `b=fir1(n,$W_n$)`, avec W_n le rapport entre la fréquence de coupure et la fréquence maximale d'observation du signal.W_n est un réel compris entre 0 et 1. n est le degré du polynôme B. Le filtre sera dit d'ordre n.

Dans la transformée en z, les hautes fréquences correspondent aux petits détails tandis que les basses fréquences donnent l'allure générale de l'image. Ainsi, un filtre passe-bas laissant passer les basses fréquences et supprimant les hautes fréquences donc les détails, va flouter l'image. En revanche, un filtre passe-haut aura tendance à devoiler les détails de l'image donnant la sensation qu'elle est plus nette.

On s'interesse ici au filtre passe-bas.

Voici le code matlab utilisé :

```
>> b=fir1(n,Wn);
>> x=reshape(IM,1,size(IM,1)*size(IM,2));
>> x=double(x);
>> y=filter(b,1,x);
```

```
>> yfinal=reshape(y,size(IM,1),size(IM,2));
>> image(yfinal);
```

Résultat :"

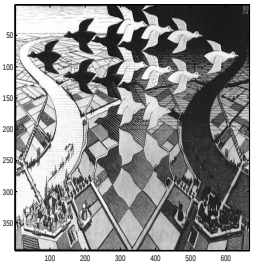
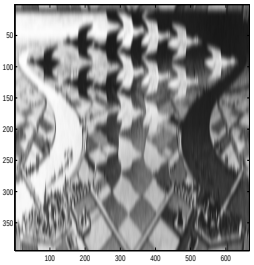



image initiale image filtrée avec n=20 et Wn=0.01

L'effet obtenu par ce filtre est un floutage de l'image.
On remarque que plus n est grand, plus le polynome représentant le filtre est précis, i.e. si W_n est petit il faudra un polynome de degré élevé pour réussir à obtenir le flou voulu.
On remarque aussi un decalage vertical de l'image que l'on expliquera et corrigera par la suite.

1.1.2 Phase non linéaire : Butterworth

Dans le cas du filtre de Butterworth, A n'est plus constant, on a donc un filtre à réponse impulsionnelle infinie.
Commande Matlab utilisée pour l'obtention de A et B :

```
[B,A]=butter(n,Wn);
```

Résultat :

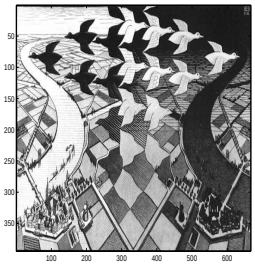
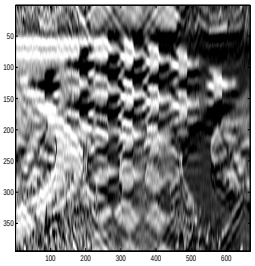



image initiale image filtrée avec n=20 et Wn=0.1

1.2 Filtres passe-haut à phase linéaire

Le filtre passe-haut est un filtre laissant passer les hautes fréquences et supprimant les basses fréquences. On génère les coefficients de ce filtre à l'aide de la commande `fir1(n,Wn,'high')` .
On obtient ainsi une image plus nette, où les éléments trops gros ont été supprimés et où seuls les détails restent présents. Avec Wn=0.1 et n=20 , on obtient :

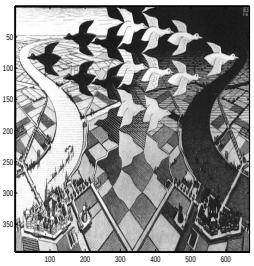
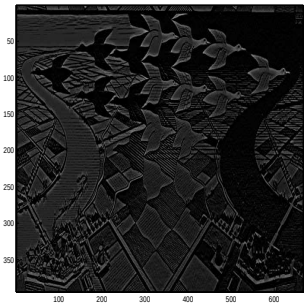



image initiale image filtrée avec n=20 et Wn=0.1

Plus on diminue Wn, plus on laisse passer d'information, et donc plus l'image ressemble à l'image originale.

En revanche, si on augmente Wn, on ne voit plus que les contours. De plus l'image s'assombrit de plus en plus, ce qui sera corrigé en augmentant le contraste et la luminosité de telle sorte que les indices restent entre 0 et 255. Pour cela, on change les couleurs par la commande `Matlab` : $y = (y - min(y))/(max(y) - min(y)) * 255;$, puis on modifie le contraste : on met à 0 en dessous d'un certain seuil (disons 127 par exemple) et 255 au dessus de ce seuil : $y = (y > 127) * 255;$.

On obtient les résultats suivants :

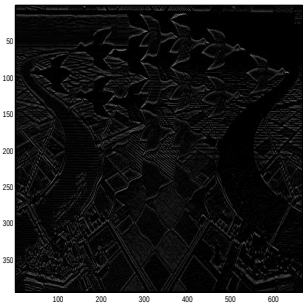
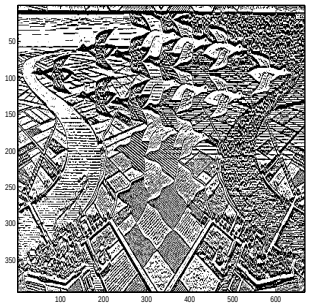



image filtrée avec n=20 et Wn=0.15 image filtrée avec correction des couleurs

On détermine enfin le k pour corriger le décalage vertical de l'image. On obtient dans ce cas 11. On sait donc que l'image a été décalée de 11 pixels verticalement. On opère donc à la translation inverse pour obtenir l'image finale :

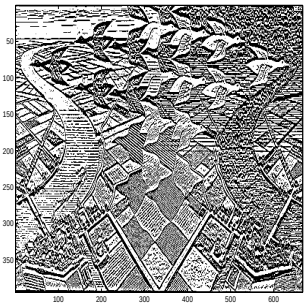


image filtrée avec correction des couleurs et du decalage

1.3 Décimation-interpolation

La décimation consiste à supprimer un pixel sur deux de l'image. L'interpolation consiste à intercaler entre chaque pixel un pixel noir (de valeur égale à 0). Dans le cas présent, nous effectuons une décimation suivie immédiatement d'une interpolation, ce qui revient à remplacer un pixel sur deux par un pixel noir.

L'image obtenue après ces deux opération est une image de même taille rayée verticalement de noir.

On applique ensuite un filtre passe-bas. Comme ce filtre élimine les hautes fréquences, et qu'une rayure noire d'un seul pixel est un élément de haute fréquence, toutes les rayures sont supprimées et l'on obtient l'image originale, plus sombre puisque l'énergie totale de l'image a été divisée par deux, et un peu plus floue (les autres détails de haute fréquence ont été aussi supprimés). On peut alors réajuster la

luminosité et le contraste de l'image pour obtenir un image plus claire, de valeurs comprises entre 0 et 255, ressemblant plus à l'image originale.

On s'aperçoit que cette opération nous a permis de diviser par deux la taille de l'image (puisqu'un pixel sur deux est noir, ces pixels n'ont pas besoin d'être stockés), sans trop perdre de qualité : il s'agit d'une compression un peu naïve et peu efficace mais qu'on pourra améliorer en itérant ces opération.

Conclusion de la partie 1

Nous avons pu programmer une compression avec perte d'information mais de qualité honorable, permettant de diviser par deux le nombre d'informations à stocker. Cette compression utilisant un filtre passe-bas chargé d'éliminer les pixels noirs insérés, elle élimine tous les détails d'un seul pixel.

2 Banc de filtres et reconstruction parfaite

2.1 Principe

Le principe que nous allons mettre en oeuvre est de décomposer l'image en 2 domaines de fréquences rendant l'image parfaitement reconstructible à partir des 2 sous-images. L'opération de filtrage passe-haut et passe-bas doublant le volume d'informations, on pourra effectuer une décimation-interpolation pour garder le même nombre d'informations que dans l'image initiale. Cette reconstruction sera parfaite sous la seule condition que les filtres H_0 et H_1 (les filtres passe bas et passe haut initiaux) soient complémentaires, c'est à dire :

$$\begin{cases} G_0 = (z) = H_1(-z) \\ G_1(z) = -H_0(-z) \\ H_0(z)G_0(z) + H_1(z)G_1(z) = Cz^{-k} \end{cases} \quad (1)$$

Le signal de sortie y sera alors égal au signal d'entrée x, à un retard k et un facteur d'amplification $\frac{C}{2}$ près. On a un banc de filtre à reconstitution parfaite, c'est à dire sans compression.

2.2 Décomposition de l'image

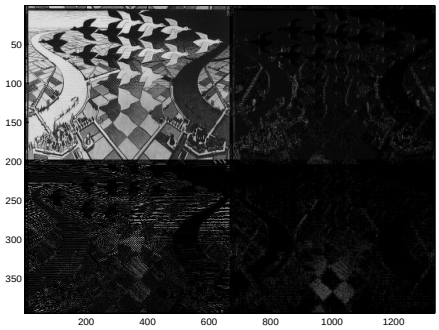
Puisque nous sommes en 2D, nous allons diviser l'image en 4 parties. Nous obtiendrons à la fin une image globale de même taille qu'au début composée de 4 petites images. Notre but est de conserver le plus d'information possible dans l'une des 4 et queles autres en contiennent peu. Nous noterons ces images comme suit :

x_{00}	x_{01}
x_{10}	x_{11}

Le banc de filtre utilisé comporte 2 couples de filtres, tous de types RIF, l'un passe-bas H_0 et son complémentaire H_1 un passe-haut, ainsi qu'un passe-bas : G_0 et un autre passe-haut : G_1. On aplique le schéma vu en TP et on obtient :

- x_{00} , qui a subi deux filtrages passe-bas, l'un sur ses lignes, l'autres sur ses colonnes. N'ayant subi que des passe-bas, cette image reste très proche visuellement de l'image initiale.
- x_{01}, qui a subi un passe-bas sur les lignes et un passe-haut sur les colonnes. Elle nous donne donc l'information, manquante à x_{00}, des contours sur les colonnes.
- x_{10}„ où on a effectué le même traitement que pour x_{01} mais sur les lignes.
- x_{11} n'a subi que des passe-haut, elle est donc la plus éloignée visuellement de l'image de base. Elle garde de l'information sur les contours mal traités par les deux précédentes, comme les contours diagonaux par exemple.

Chaque sous image a été décimée deux fois, la première lors du filtrage sur les lignes, puis lors du filtrage sur les colonnes. Chaque sous image a donc 4 fois moins d'information que l'image initiale, et on retrouve donc la même information en les ajoutant pour obtenir l'image finale.



Visualisation des quatre sous-images

On voit que chacune des sous-images a un décalage vertical et horizontal : pour connaitre le nombre exact de pixels, on peut appliquer les filtres utilisés à un dirac monodimensionnel, et ainsi observer la réponse impulsionnelle, et donc le nombre de pixels dont celle-ci est décalée.
On a aussi dû appliquer une transformation affine pour 3 des sous-images (toutes sauf $x_0 0$) afin que les valeurs des pixels soient comprises entre 0 et 255.

2.3 Recomposition de l'image

On reconstruit ensuite le signal original en appliquant les filtres G_0 et G_1 sur les colonnes puis les lignes (= dans l'ordre inverse des filtres H_0 et H_1). Ces filtres nous donnent une reconstruction parfaite du moment où ils verifient la relation de complémentarité vue précédemment.
On vérifie cela en calculant l'erreur entre l'image originale et l'image reconstruite : on obtient alors une erreur de l'ordre du zéro machine, ce qui confirme qu'il n'y a pas de pertes d'informations.
Par ailleurs, on a aussi appliqué les filtres H_0 et G_0 à un dirac monodimensionnel, et on a ainsi pu observer que le maximum du signal observé a été translaté de 23 pixels par rapport au maximum du dirac (en x=0).

Conclusion de la partie 2

Les transformations appliquées à notre image a permis de la décomposer en sous images dont certaines comportaient de l'information facilement compressible par des algorithmes comme gzip. C'est ce que l'on fera par la suite en itérant ces transformations.

3 Compression des images

3.1 Principe

On itère sur la première image x_{00} la décomposition en images hautes et basses fréquences. Les images basses fréquences contiendront beaucoup de valeurs semblables et les hautes fréquences beaucoup de zéros (à moins d'avoir une image extremement bruitée, ce qui n'est pas le cas ici !) et seront donc déjà facilement compressibles par gzip (en effet, gzip identifie les suites d'éléments consécutifs égaux pour les réduire à 2 éléments : la valeur, et le nombre de fois où elle est répétée).
De plus, les images x_{00}^i contiennent l'essentiel de l'information visuelle (moyenne de l'image et faibles variations) donc on ne la quantifiera pas mais on itérera les decompositions sur elles.
C'est donc les images hautes fréquences que l'on essaira de quantifier le plus pour les rendre plus facilement compressibles par gzip. Les images x_{01} et x_{10} contiennent les détails horizontaux ou verticaux, tandis que x_{11} contient les details à la fois verticaux et horizontaux (donc diagonaux) puisqu'il a subit un passe-haut dans les deux directions.

3.2 Résultats : Optimisation d'un facteur 10

On cherche à obtenir un facteur de compression de 10 (rapport entre l'image directement compressée par gzip, et l'ensemble des sous images sauvegardé puis compressé avec gzip). Pour cela, on se fiera à un

résultat visuel de l'image plutot qu'à un calcul d'erreur, puisque le but est que l'image reste esthétiquement correcte.
Les coefficients retenus pour la quantification de chaque sous image sont :

```
q=[1 800 800 300; 1 200 200 150; 1 100 100 50; 1 35 35 12]
```

On obtient ainsi une erreur relative d'environ 17% en comparant l'image compressée et l'image originale, donc une perte de 17% d'informations.

Résultat quantitatif de la compression :

```
-rw-r--r--    1 jgprince etud_apo    25032 Dec 16 18:25 compresse.mat.gz
-rw-r--r--    1 jgprince etud_apo   242664 Dec  3 15:53 escher.mat.gz
-rw-r--r--    1 jgprince etud_apo    72984 Dec 16 20:33 escher.jpg
```

soit un rapport de 1/10 environ (meilleur qu'un jpg sans pertes...).

Conclusion

Au cours de ce TP, nous avons dans un premier temps étudié différents filtres : passe-bas, passe-haut, à phase linéaire ou non, ... et observé leurs effets sur des signaux monodimensionnels.
Puis on a appliqué un banc de filtre à notre image, signal bidimensionnel, au cours de la seconde partie.
Enfin, on a pu observer les effets qualitatif et quantitatifs d'une compression d'image. Le but, arriver à une bonne compression (facteur 10) sans trop altérer la qualité visuelle de l'image a été atteint.