Compression par transformée, Jpeg et Jpeg 2000

Ch. Dossal

Novembre 2008

1 Introduction

Il existe un large éventail d'algorithmes de compression d'images. Nous proposons ici de nous intéresser à deux codeurs en particulier que sont jpeg et jpeg 2000. Ce TD propose de réaliser sous matlab certaines parties de ces codeurs.
Il est possible de comprimer des signaux sans perte en utilisant par exemple des codages type Huffman ou arithmétique. Ces codages exploitent le fait que la fréquence d'apparition de chaque symbole à coder est différente. Ainsi on exploite le fait que certaines configurations sont plus fréquentes que d'autres et qu'il est plus intelligent de coder les configurations fréquentes avec moins de bits que celles qui sont rares.
Les méthodes de compression d'image exploitent en plus la redondance qui existe au sein d'une image. Les valeurs des niveaux de gris de deux pixels proches sont souvent proches. Il parait donc intéressant de coder ces valeurs ensemble. Un aspect central de la compression d'image est d'exploiter une représentation qui sépare l'image en données indépendantes. Ces données seront des coeffcients de cosinus locaux dans Jpeg et et des coefficeints d'ondelettes dans Jpeg2000.
Nous traiterons dans ce TD

- des transformées en cosinus locaux et en ondelettes,
- de la quantification,
- du nombre de bits nécessaires à coder ces transformées,
- et de la relation nombre de bits vs distorsion.

Nous traiterons uniquement des images noir et blanc et n'aborderons pas les changements d'espaces de coleurs RVB vers YUV par exemple.
D'autre part nous n'aborderons pas tous les aspects de codage contextuels qui font la puissance et l'efficacité de jpeg2000.

2 Compression Jpeg

Le format d'image jpeg utilise une tansformée en cosinus locaux discrets (DCT). Cette transformée est une variante réelle de la fft locale. La commande matlab qui permet d'effectuer cette DCT sur des tableaux 2D est $dct2$. Cette dct est effectuée sur des blocks 8×8.

1. Ecrire un programme qui prend pour entrée une image sous forme d'un tableau $N \times N$ de réels, pù N est un multiple de 8 et qui renvoie le tableau $N \times N$ des coefficients de DCT locale.

```
function C=DCTGlobale(I)
```

2. Visualiser cette DCT sur plusieurs exemples d'images. Qu'observez vous ?

3. Ecrire le programme inverse qui reconstruit une image à partir d'un tableau de coefficients. Vérifiez que la reconstruction est parfaite si on ne modifie pas la DCT sur quelques exemples.

4. Après avoir décomposé l'image dans une DCT locale on doit quantifier les coefficients obtenus. Il existe plusieurs moyens de quantifier les données. Nous utiliserons un quantificateur uniforme à zone morte c'est à dire un quantificateur à pas de quantification constant sauf pour la zone associée à 0 qui sera plus grande. On prendra ici 2 fois plus grande. La zone 0 joue ici un rôle particulier dans la mesure où la DCT a permi normalement de mettre un grand nombre de coefficients à 0.
   Ecrire un programme qui quantifie un tableau de nombre quelconque avec un pas de quantification q avec une zone correpondant à la valeur 0 symétique de taille 2q.

   ```
   function Q=Quantification(C,q)
   ```

   On peut se contenter d'une ligne avec la commande $f$ix.
   En pratique jpeg utilise des pas de quantification variables selon les 64 bandes de fréqences. Nous nous contenterons d'un pas de quantifiaction unique.

5. Ecrire un programme qui affiche l'histogramme des coefficients quantifiés pour plusieurs pas de quantification et qui calcule l'entropie et le nombre de bits théorique nécessaire au codage.

6. Ecrire un programme qui calcule et affiche la courbe $d$istorsion vs nombre de bits par pixels. La distorsion est mesurée à l'aide du PSNR.

7. Faire la même chose non plus sur la DCT mais sur l'image originale. Comparer les deux courbes.

3 Compression Jpeg2000

Le standart de compression Jpeg2000 est une norme de compression qui utilise les ondelettes 2D. En pratique ce sont des ondelettes biorthogonales 7-9 qui sont le plus souvent utilisées. Les performances de jpeg2000 réside en partie dans l'utilisation des ondelettes mais est également dû à un codage contextuel des coefficients extrèmement performant. Ce dernier point très technique ne sera pas abordé ici.

8. Ecrire un programme qui calcule la transformée en ondelettes de 7-9 d'une image et qui la quantifie

   ```
   function Q=QuantOnd(I,q)
   ```

   On rappelle que la transformée en ondelettes biorthogonale s'effectue avec la commande

   ```
   WB=FWT2_PB(I,L,qmf,dqmf)
   ```

   et que les filtres deux filtres d'analyse et de synthèse peuvent être obtenus par la commande

```
[qmf,dqmf] = MakeBSFilter('Villasenor',1);
```

9. Avec le programme réalisé dans la section précédente, visulaliser l'histogramme des coefficients.
10. Ecrire un programme qui calcule et affiche la courbe *d*istorsion vs nombre de bits par pixels
11. Comparer ces résultats avec ceux obtenus pour la DCT.