TP5 : Analyse espace-fréquence pour les images : la DCT.

Fabien PIERRE
fabien.pierre@math.u-bordeaux1.fr
http://sites.google.com/site/fabienpierre/enseignements

Automne 2013.

1 La DCT.

La DCT est une transformation linéaire, qui correspond à l'intégration du signal contre des cosinus discrets. Elle s'apparente à la DFT avec l'avantage de mieux rassembler l'énergie des basses fréquences sur 0.

- Tracer la DFT et la DCT (commande `dct2`) de l'image Lena. Comparer.
- Effectuer un zoom par zéro-padding en domaine DCT.
- Ajouter un bruit centré gaussien à l'image. Transformer l'image bruitée dans le domaine domaine DCT. Observer.
- Débruiter l'image bruitée en effectuant un seuillage doux dans le domaine DCT. Comparer les résultats pour divers seuils et niveaux de bruits. Essayer avec un seuillage dur.

2 La DCT par block.

Dans cette section, on introduit une nouvelle transformation. Afin de tenir compte des variations des propriétés statistiques de l'image, on effectuera la transformée DCT sur des blocs de taille 8×8.

- À l'aide de boucles `for`, créer la fonction `block_dct` qui prend en entrée une image et retourne la DCT sur des blocs de taille 8×8.
- À l'aide de boucles `for`, créez la fonction `block_idct` qui est l'inverse de la fonction précédente.
- Optimiser votre code à l'aide de la commande `blockproc`. Comparer les temps de calcul.

Exemple :

```
fun = @(block_struct) dct2(block_struct.data);
DCT = blockproc(I,[8 8],fun);
```

- Tracer la DCT par bloc de l'image Lena.
- Ajouter un bruit centré gaussien à l'image. Transformer l'image bruitée en domaine DCT par bloc. Observer.
- Débruiter l'image bruitée en effectuant un seuillage doux dans le domaine DCT par bloc. Comparer les résultats pour divers seuils et niveaux de bruits.

3 Codage JPEG.

Le but de cette section ne sera pas d'implémenter un codeur JPEG mais de comprendre le fonctionnement du système et notamment la partie analyse.

Rappelons la chaine de codage JPEG :

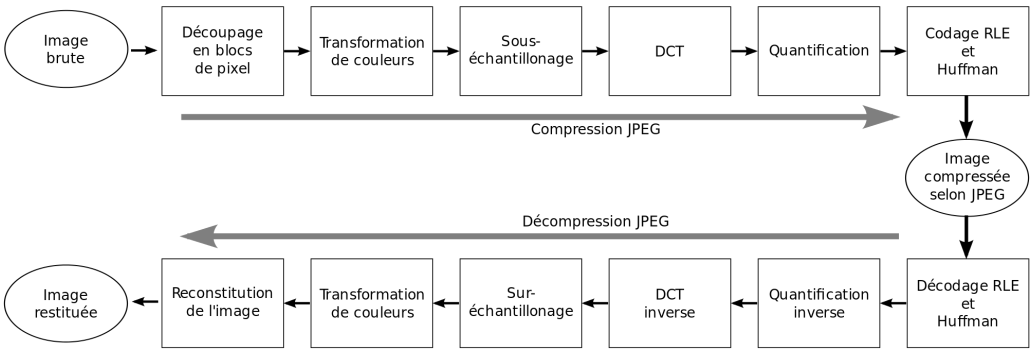

FIGURE 1 – Schéma de bloc du codage JPEG couleur.

Commençons par rappeler que l'entropie, en informatique, est une donnée permettant de calculer la quantité d'information. La quantité théorique de bits minimale nécessaire pour envoyer un information d'entropie e au travars un canal parfait (sans perte) est donnée par :

$$I = \lceil -e \rceil.$$

Le codage de Huffman permet d'approcher cette borne inf. Afin de diminuer au maximum le quantité d'information à transmettre, il faut donc diminuer l'entropie de l'information à transmettre. Rappelons que l'entropie d'une variable aléatoire suivant une loi de probabilité discre $(\mu_1, \mu_2, ..., \mu_n)$ est donnée par

$$e = -\sum_{i=1}^{n} \mu_i \log_2(\mu_i).$$

Ainsi, l'entropie sera maximale si la distribution est uniforme, et minimle si la distribution est constante. Ce qui se conforme facilement à l'intuition : il sera plus facile de tranmettre une image constante (il suffit de donner la constante) qu'un brruit blanc gaussien.

- Comparer l'entropie (commande `entropy`) de l'image lena et de l'image lena à laquelle vous aurez ajouté un bruit gaussien, de variance assez forte.
- Comparer l'image de lena avec l'image de lena quantifiée.
- Comparer l'entropie de l'image lena avec l'entropie de la dct par bloc de l'image lena.
- Quantifier la dct par bloc de l'image lena par une quantification à zone morte (utiliser la commande `fix`). Comparer l'entropie de cette quantification avec l'entropie de l'image lena. Recommencer pour plusieurs pas de quantification.
- Visualiser l'image reconstruite à partir de la dct quantifiée.

4 Inpainting en DCT.

Nous n'aborderaons pas en détail la théorie permettant de justifier la contruction de cet algorithme, nous donnons seulemtn ici une utilisation de la DCT par bloc.

L'inpainting vise à retrouver des données perdues d'une image. On peut justifier heuristiquement cet algorithme en remarquant que la dct par bloc contient beaucoup de zéros, il suffit alors de tronquer la dct pour éliminer les petites composante et ne conserver que les plus grandes.

On suppose que l'on a l'image masquée i.e. l'image donnée par l'opérateur de masquage défini comme suit :

$$M(I_{i,j}) = \begin{cases} I_{i,j} & \text{si l'image est non masquée,} \\ 0 & \text{sinon.} \end{cases}$$

L'algorithme est le suivant :

Avec $A^t = blockDct.M$ et $A = M.blockIdct$, M l'opérateur de masquage et $SD[x, \gamma]$ le seuillage doux de x par le seuil γ.

Algorithme 1 Iterative soft thresholding.

$\gamma > 0$
$x_0 \leftarrow$ Image masquée.
for $n > 0$ **do**
    $x_n \leftarrow SD\left[x_n + A^t\left(y - Ax_n\right), \gamma/2\right]$
end for
Résultat $= Ax_n$

– Créez le masque et l'image masquée en masquant 70% des valeurs de l'image aléatoirement.
– Mettez en place l'algorithme de seuillage doux itéré.

NB : Vus pouvez encore optimiser votre code en utilisant la toolbox DCT téléchargeable sur la page :

`https://sites.google.com/site/fabienpierre/a`