Analyse par ondelettes

Travaux pratiques
Feuille 4 :
Paquets d'ondelettes et parcimonie

Exercice 1 Préparation du TP

1. Téléchargement et compilation des programmes Megawave :
   - Télécharger (et sauvegarder dans votre my_megawave2/src) le fichier *tp4.tgz*. Il se trouve sur la page
     http ://www.math.univ-toulouse.fr/∼fmalgouy/
     rubrique "enseignement", puis "Analyse par ondelettes".
   - Aller dans votre compte Megawave
     cd $MY_MEGAWAVE2/src/
   - Décompresser l'archive
     tar xvzf tp4.tgz
   - Compiler les modules Megawave
     cmw2_all TP4/
2. Pour voir le code correspondant aux algorithmes que vous utiliserez, il faut ouvrir les fichiers *C* :
   gedit TP4/.c &*
3. Préparation des résultats :
   - création d'un répertoire :
     mkdir resultats_tp4
   - déplacement des données sur lesquelles on fera les expériences :
     cp TP4/barbara.gif resultats_tp4/
     cp $MEGAWAVE2/data/wave/packets/ortho/da04.ir resultats_tp4/
   - Changement de répertoire :
     cd resultats_tp4
   - Lancement de commandes Megawave :
     wp2dmktree -w 1 arbre
     wp2dview -R decomp_1.img arbre barbara.gif da04.ir
   - Visualisation du résultat :
     xv decomp_1.img &

4. Utilisation de scripts pour gagner du temps et pouvoir refaire un exercice :
   - Ouverture du fichier contenant le script *bash*
     gedit correction_tp4.sh &
   - Écriture de l'entête du fichier *bash*
     Écrire sur la première ligne du fichier :
     # !/bin/bash
   - Utilisation du fichier *bash* à l'aide de copier-coller
     Dans les exercices suivants : Faire un copier-coller, depuis le terminal vers le fichier pour mémoriser les

commandes :
wp2dmktree -w 1 arbre
wp2dview -R decomp_1.img arbre barbara.gif da04.ir

– Pour lancer le fichier *bash* dans le terminal, il faut modifier ses droits :
chmod 700 correction_tp4.sh
Vous pourrez ensuite l'executer :
./correction_tp4.sh

Exercice 2 Visualisation des paquets d'ondelettes

1. Dans cette question, on va visualiser la décomposition en paquets d'ondelettes d'une image, pour un arbre complètement décomposé et différents niveaux de décomposition. Les commandes suivantes permettent de visualiser la transformée en ondelette pour les niveau 1, 2, 3 et 6, pour l'ondelette de Daubechies 4:
wp2dmktree -f 1 arbre
wp2dview arbre barbara.img da04.ir
wp2dmktree -f 2 arbre
wp2dview arbre barbara.img da04.ir
wp2dmktree -f 3 arbre
wp2dview arbre barbara.img da04.ir
wp2dmktree -f 6 arbre
wp2dview arbre barbara.img da04.ir

2. Dans cette question, nous allons visualiser les différents éléments d'une base de paquets d'ondelettes. Lancer les commandes suivantes et pour chaque question commenter le résultat (forme des éléments, taille du support,...)
   (a) Lancer la commande:
   voir_base_paquets_ondelettes 128 da04.ir 3 paquets
   (b) Pour visualiser les éléments de la base:
   xv paquets_* &
   Vous pouvez ensuite changer d'image (un clique à droite, puis clique sur le nom de l'image).
3. Dans cette question, nous allons visualiser les transformée de Fourier des différents éléments d'une base de paquets d'ondelettes. Afin de les afficher des plus basses aux plus hautes fréquences, nous devons afficher le paquet $p = G^{-1}(k) + 1$, pour $k = 0, \dots 7$ (la fonction G est définie dans le cours). On doit donc lancer les commandes:
   (a) *voir_fft -p 1 paquets_01*
   (b) *voir_fft -p 1 paquets_02*
   (c) *voir_fft -p 1 paquets_04*
   (d) *voir_fft -p 1 paquets_03*
   (e) *voir_fft -p 1 paquets_07*
   (f) *voir_fft -p 1 paquets_08*
   (g) *voir_fft -p 1 paquets_06*
   (h) *voir_fft -p 1 paquets_05*

Exercice 3 Débruitage avec un dictionaire de paquets d'ondelettes

On considère le cas où l'on a un bruit additif Gaussien sur une image:

$$v = u + b, \tag{1}$$

où u est l'image idéale, b le bruit et v la donnée dont on dispose. Quelque-soit la base, les coordonnées de v sont la somme des coordonnées de u et de b. Dans le TP2, on a vu qu'une méthode pour enlever le bruit de v consiste à seuiller ses coefficients d'ondelettes.

Une variation simple autour de cette idée utilise la remarque suivante: "rechercher l'image est équivalent à chercher le bruit". En effet, ils sont reliés par (1) et on connait v. Comme dans le TP 2, on va utiliser le fait que dans une base (et même un dictionnaire) les coordonnées d'un bruit Gaussien sont souvent faible.

Formellement, si l'on considère une base (on peut l'étendre à un dictionnaire) consituée d'éléments $(\psi_i)_{i\in I}$, on a pour τ assez grand que

$$\mathbb{P}\left(b_{\psi_i} \leq \tau\right) \text{ est très forte, pour tout } i \in I,$$

où b_{ψ_i} est une coordonnée de b dans $(\psi_i)_{i\in I}$.

On peut donc construire un algorithme recherchant le bruit. Il consistera a effacer l'information successivement dans l'ensemble des directions ψ_i. Un tel algorithme est décrit dans le Tableau 1.

* Entrées : l'image bruitée v, un paramètre réel $\tau > 0$, un dictionaire ou une base $(\psi_i)_{i\in I}$
* Sortie : l'image résultat u^*
* L'algorithme :
  - Initialisation $b = v$
  - Pour chaque $i \in I$

$$b_{\psi_i} = \begin{cases} b_{\psi_i} & \text{, si } |b_{\psi_i}| < \tau \\ \tau & \text{, si } b_{\psi_i} \geq \tau \\ -\tau & \text{, si } b_{\psi_i} \leq -\tau \end{cases}$$

  - $u^* = v - b$.

TABLE 1 – Un algorithme de débruitage qui cherche le bruit, puis l'enlève de l'image. Il peut être appliqué à un dictionaire de paquets d'ondelettes.

1. Utiliser la fonction *fnoise* pour construire une version de *barbara.gif* bruitée par un bruit blanc Gaussien d'écart type 10.
2. Pour appliquer l'algorithme décrit dans la Table 1 à une image bruitée *bruitee*, vous pouvez utiliser la commande
   wp2doperate -t 2 -S 20 -L 4 tree da04.ir bruitee resultat
   où $\tau = 20$, *tree* est un arbre compètement décomposé de niveau 4, *da04.ir* est l'ondelette de Daubechies 4, *bruitee* est l'image bruite et *resultat* est l'image résultat.
   Mettre dans un fichier les commandes pour faire une experience de débruitage similaire à celle que l'on a faite dans les TP précédents.
3. Comparer les meilleurs résultats obtenus à la question précédente avec les meilleurs résultats des TP précédents.

Exercice 4 Vers le Compressed Sensing

Le "Compressed Sensing" (voir [1, 2, 3]) désigne un ensemble de résultats mettant en évidence le fait que le "Basis Pursuit Denoising" permet de reconstruire avec beaucoup de précision des coordonnées $(\lambda_i)_{i\in I}$ à partir d'une mesure $\sum_{i\in I}\lambda_i\psi_i$ (typiquement $\#I >> N$). Typiquement, la reconstruction est très bonne lorsque les $(\psi_i)_{i\in I}$ sont bien choisis et que la donnée $(\lambda_i)_{i\in I}$ de départ a peu de coordonnées non nulles (on dit alors qu'elle est parcimonieuse ("sparse", en anglais)).

Le modèle appelé "Basis Pursuit Denoising" prend la forme

$$(BP) : \min_{(\lambda_i)_{i\in I}\in\mathbb{R}^I} \|\sum_{i\in I}\lambda_i\psi_i - v\|^2 + \lambda\sum_{i\in I}|\lambda_i|,$$

où $\lambda > 0$ *est un paramètre et* $\|.\|$ *représente la norme euclidienne.*

Le but de cet exercice est de faire une des expériences étonnantes à l'origine de ces travaux.

1. *Détailler le fonctionnement des modules* randomImage.c, fmse_coordinates.c, bp_wp_DDD.c *et du script* CS_script.
2. *En considérant l'image* randImg *et le dictionnaire utilisés dans le script* CS_script*, quelle est la taille du système (nombre d'inconnues et d'équations)*

   $$\sum_{i\in I}\lambda_i\psi_i = randImg\ ?$$

   Ce système est-il inversible?
3. *Lancer le script* CS_script *et commenter les différents résultats obtenus.*

Références

[1] E. Candes, J. Romberg, and T. Tao. Robust uncertainty principles : Exact signal reconstruction from highly incomplete frequency information. *IEEE, Trans. on Information Theory*, 52(2) :489–509, Feb. 2006.

[2] D. Donoho. Compressed sensing. *IEEE, Trans. on Information Theory*, 52(4) :1289–1306, April 2006.

[3] Compressed Sensing ressources. http ://www.dsp.ece.rice.edu/cs/.