Analyse par ondelettes

Travaux pratiques
Feuille 3 :
Transformée en ondelettes et paquets d'ondelettes

Exercice 1 Préparation du TP

1. Téléchargement et compilation des programmes Megawave :
   - Télécharger (et sauvegarder dans votre my_megawave2/src) le fichier *tp3.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 tp3.tgz
   - Compiler les modules Megawave
     cmw2_all TP3/
2. Pour voir le code correspondant aux algorithmes que vous utiliserez, il faut ouvrir les fichiers *C* :
   gedit TP3/.c &*
3. Préparation des résultats :
   - création d'un répertoire :
     mkdir resultats_tp3
   - déplacement des données sur lesquelles on fera les expériences :
     cp TP3/barbara.gif resultats_tp3/
     cp $MEGAWAVE2/data/wave/packets/ortho/da04.ir resultats_tp3/
   - Changement de répertoire :
     cd resultats_tp3
   - 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_tp3.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_tp3.sh
Vous pourrez ensuite l'executer :
./correction_tp3.sh

Exercice 2 Compression

Lorsque l'on considère une façon de représenter les images (ici on les représente par leurs coefficients d'ondelettes), l'une des propriétés très importante est la possibilité de représenter (ou d'approximer) une image avec peu de coefficients. Par exemple, en compression d'images, on ne garde (et ne code) que les grands coefficients, les autres étant mis à zéro. Ainsi, si l'on veut compresser une image $u \in \mathbb{R}^{N^2}$ avec la base d'ondelette $\mathcal{B}$ (comme le fait le standard de compression d'images "JPEG 2000"), on ne code que les coefficients dont le module est supérieur à une valeur $\tau > 0$ donnée. Les coordonnées $(c'_\psi)_{\psi\in\mathcal{B}}$ de l'image u' ainsi obtenue sont donc définies pas

$$c'_\psi = \begin{cases} c_\psi & \text{, si } |c_\psi| \geq \tau, \\ 0 & \text{, sinon,} \end{cases}$$

où $(c_\psi)_{\psi\in\mathcal{B}}$ sont les coordonnées de u dans la base d'ondelettes $\mathcal{B}$.

1. Lire le module Megawave *seuille_ondelettes.c* et dire ce qu'il fait.
2. Utiliser le module *seuille_ondelettes* pour les valeurs de $\tau = 1, 5, 25, 125$ pour construire les image u', à partir de l'image *barbara.gif*. Vous considérerez une base de niveau 4 construite avec l'ondelette de Daubechies 4.
   Visualiser ces images et comparer les résultats obtenus avec ceux de l'exercice analogue du TP sur la transformée de Fourier.
   Visualiser les coefficients d'ondelettes des résultats avec les commandes (ci-dessous, il faut remplacer *resultat* par le nom de l'image dont vous voulez voir les coefficients d'ondelettes)
   wp2dmktree -w 4 arbre
   wp2dview -s arbre resultat da04.ir

   Dire comment se comporte
   $$C(\tau) = \#\{\psi \in \mathcal{B}, |c_\psi| \geq \tau\}$$
   (où # désigne le cardinal d'un ensemble) et la qualité de l'image obtenue lorsque τ augmente.
   Remarque : La quantité $C(\tau)$ est importante puisqu'elle correspond au nombre de coefficients que l'on doit coder pour représenter u'. Ainsi elle correspond (grosso modo) à la longueur du code.
3. Ajuster τ de manière à approximer l'image *barbara.gif* en ne gardant que 10% de ses coefficients d'ondelettes. Visualiser et décrire le résultat. Comparer le au résultat de l'exercice analogue du TP sur la transformée de Fourier.
4. Montrer que si $\tau \leq \tau'$
   $$C(\tau') \leq C(\tau)$$
   et que $l^2(\tau) \leq l^2(\tau')$ avec, pour tout $t \geq 0$
   $$l^2(t) = \left(\frac{1}{N^2} \sum_{\psi, |c_\psi| < t} |c_\psi|^2 \right)^{\frac{1}{2}}.$$
   Remarque : $l^2(\tau)$ quantifie l'erreur que l'on fait en approximant u par u'.
5. Pour une valeur de $\tau \geq 0$, on obtient une valeur $\#C(\tau)$ et une valeur $l^2(\tau)$. Si l'on fait varier τ, on obtient une courbe. Plus cette courbe décroît rapidement plus la façon de représenter les images (ici les ondelettes) est efficace en compression d'images (on a une erreur faible bien que l'on code peu de coefficients). Le but de cette question est de représenter cette courbe.

(a) Lire le module Megawave *courbe_seuillage_ondelettes.c* et dire ce qu'il fait.

(b) Lancer la commande :
courbe_seuillage_ondelettes barbara.gif da04.ir 4 courbe
pour construire la courbe donnant l'erreur en fonction du nombre de coefficients que l'on doit coder.

(c) Visualiser (avec *splot*) et commenter la courbe obtenue à la question précédente. La comparer à la courbe obtenue pour une autre base, si vous en avez une.

Exercice 3 Débruitage

La propriété vue ci-dessus pour la compression d'image est aussi importante en débruitage d'images. En effet, il n'est pas très difficile de voir que les coordonnées dans une base orthogonale d'ondelettes d'une image ne contenant que du bruit Gaussien est un bruit Gaussien de même écart type. Ainsi, ces coordonnées seront typiquement petites et réparties sur tous les ψ de notre base d'ondelettes.

Lorsque l'on a un bruit additif Gaussien sur une image

$$v = u + b,$$

où u est l'image idéale, b le bruit et v la donnée dont on dispose, les coordonnées de v sont la somme des coordonnées de u et de b. Ainsi, si seulement quelques-unes des coordonnées de u sont grandes (et les autres sont nulles). Annuler les petites coordonnées de v permet de retrouver des coordonnées très proche de celles de u.

C'est ce principe que nous allons mettre en évidence dans ce TP.

1. **Transformée en ondelette d'un bruit.** Utiliser les commandes suivantes pour créer une image de bruit, visualiser le bruit et visualiser ses coordonnées dans une base d'ondelette de niveau 4 avec l'ondelette de Daubechies 4.
   fop -a 0.0 -t barbara.gif zero
   fnoise -g 10 zero bruit
   wp2dmktree -w 4 arbre
   wp2dview arbre bruit da04.ir
   xv bruit &

   Que pensez vous des coordonnées du bruit dans la base d'ondelettes?

2. Pour débruiter l'image v, on propose de seuiller ses coordonnées. La nouvelle image u' est décrite par ses coordonnées $(c'_\psi)_{\psi\in\mathcal{B}}$:
   $$c'_\psi = \begin{cases} c_\psi & \text{, si } |c_\psi| \geq \tau, \\ 0 & \text{, sinon,} \end{cases}$$
   où $(c_\psi)_{\psi\in\mathcal{B}}$ désigne les coordonnées de v dans la base d'ondelettes.
   Utiliser les commandes suivantes pour bruiter l'image *barbara.gif* et appliquer le traitement ci-dessus (appelé "seuillage dur des coefficients d'ondelettes") pour plusieurs valeurs de τ.
   fnoise -g 10 barbara.gif bruitee
   seuille_ondelettes bruitee da04.ir 4 10 re10
   seuille_ondelettes bruitee da04.ir 4 20 re20
   seuille_ondelettes bruitee da04.ir 4 30 re30
   seuille_ondelettes bruitee da04.ir 4 40 re40
   seuille_ondelettes bruitee da04.ir 4 50 re50

3. Visualiser les images avec la commande
   xv re?0 &

(a) Que dire du bruit encore présent dans le résultat en fonction de τ (regarder les zones uniformes de l'image)?

(b) Que dire de la quantité d'information encore contenue dans le résultat, en fonction de τ (regarder notamment les yeux de barbara et la nappe)?

(c) Quelle valeur de τ préconiseriez vous, pourquoi?

Exercice 4 Visualisation des ondelettes

1. Dans cette question, on va visualiser la décomposition en ondelette d'une image, pour 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 -w 1 arbre
   wp2dview arbre barbara.gif da04.ir
   wp2dmktree -w 2 arbre
   wp2dview arbre barbara.gif da04.ir
   wp2dmktree -w 3 arbre
   wp2dview arbre barbara.gif da04.ir
   wp2dmktree -w 6 arbre
   wp2dview arbre barbara.gif da04.ir

2. Dans cette question, on va visualiser les différents éléments de la base d'ondelette. Lancer les commandes suivantes et pour chaque question commenter le résultat (forme des éléments, taille, ressemblances,...)

   (a) Lancer la commande:
   voir_base_ondelettes 64 da04.ir 3 ondel

   (b) Pour visualiser les éléments correspondants au résumé:
   xv ondel_01 &

   (c) Pour visualiser les éléments représentant les structures horizontales:
   xv ondel_02 &
   xv ondel_03 &
   xv ondel_04 &

   (d) Pour visualiser les éléments représentant les structures verticales:
   xv ondel_05 &
   xv ondel_07 &
   xv ondel_09 &

   (e) Pour visualiser les éléments représentant les structures diagonales:
   xv ondel_06 &
   xv ondel_08 &
   xv ondel_10 &

   (f) Pour visualiser les éléments à l'échelle 1 (ceux-ci sont présent toutes les translations de 2 pixels):
   xv ondel_04 &
   xv ondel_09 &
   xv ondel_10 &

(g) Pour visualiser les éléments à l'échelle 2 (ceux-ci sont présent toutes les translations de 4 pixels):
xv ondel_03 &
xv ondel_07 &
xv ondel_08 &

(h) Pour visualiser les éléments à l'échelle 3 (ceux-ci sont présent toutes les translations de 8 pixels):
xv ondel_01 &
xv ondel_02 &
xv ondel_05 &
xv ondel_06 &

Exercice 5 Déconvolution utilisant des paquets d'ondelettes

On rencontre le problème de la déconvolution dans plusieurs situations car c'est l'une des dégradations subit par l'image, lors de sa création.

Si dans un premier temps on néglige le bruit, l'équation suivante nous dit comment l'image mesuré v est relié à l'image idéale de départ u:

$$v = h * u.$$

Ci-dessus h est un noyau de convolution que l'on suppose connu.

Cette équation est parfois facile à inverser dans le domaine de Fourier. On a en effet en chaque fréquence

$$\hat{v} = \hat{h}\hat{u},$$

et donc

$$\hat{u} = \frac{\hat{v}}{\hat{h}},$$

pour les frequences où $\hat{h} \neq 0$. Par contre, aux fréquences auxquelles $\hat{h}$ est nul (ou très petit), $\hat{v}$ est nul et le contenu $\hat{u}$ de cette fréquence est perdu.

1. Éditer les fichiers *mknoyau.c* et *convol_fourier.c* et dire ce que font les fonctions qu'ils contiennent.
2. Lancer les commandes suivantes et dire ce qu'elles font:
   > *mknoyau -g 100 barbara.img filt_re*
   > *convol_fourier filt_re barbara.img floue.img*
   > *convol_fourier -i 0.001 filt_re floue.img defloue.img*
   Commenter les résultats obtenus.
3. Modifier les lignes de commandes de la question précédente pour faire une convolution suivie d'une déconvolution avec un noyau de type masque de côté 4 pixels.
4. Modifier les commandes de la question 2 pour rajouter un bruit Gaussien d'écart type 2 à l'image *floue.img* avant de la déconvoluer.
   N.B.: Vous pouvez ajouter du bruit Gaussien à l'aide de la commande *fnoise*.
5. Commenter les résultats obtenus et cherchez une valeur (entre 0 et 1) pour l'option *-i* de la fonction *convol_fourier* permettant d'obtenir les meilleurs résultats visuels.
6. Dans cette question, nous allons étudier un algorithme de déconvolution utilisant des paquets d'ondelettes. Son principe est d'approximer la convolution avec l'inverse de h dans une base de paquets d'ondelettes. Ceci est possible car les paquets d'ondelettes sont biens localisés en fréquence. Ainsi $\frac{1}{\hat{h}}$ est presque constant sur le support du paquet d'ondelette. On peut donc déconvoluer (approximativement) en multipliant par $\frac{1}{\hat{h}}$ les coefficients de paquets d'ondelettes.
   L'intérêt d'utiliser des paquest d'ondelettes plutôt que Fourier est que les paquets d'ondelettes sont mieux localisés en espace et concentrent donc mieu l'information contenue dans l'image. Le bruit et le contenu informatif de l'image sont donc mieu séparés. On peut donc les traiter différemment:

– On déconvolue l’information (i.e. les forts coefficients).
– On mets à zéro le bruit (i.e. les petits coefficients).

On peut essayer un algorithme basé sur ce principe avec les commandes suivantes:
> *mknoyau -g 100 barbara.img filt_re*
> *convol_fourier filt_re barbara.img floue.img*
> *fnoise -g 2 floue.img floue.img*
> *wp2dmktree -m 4 arbre_miroir*
> *wp2deigenval -i 0.01 filt_re arbre_miroir filt_approx*
> *wp2doperate -t 2 -S 1.3 -L 4 -c filt_approx -C 4 arbre_miroir da04.ir floue.img defloue_wp.img*

Comparez les résultats obtenus avec ceux de la question 5.