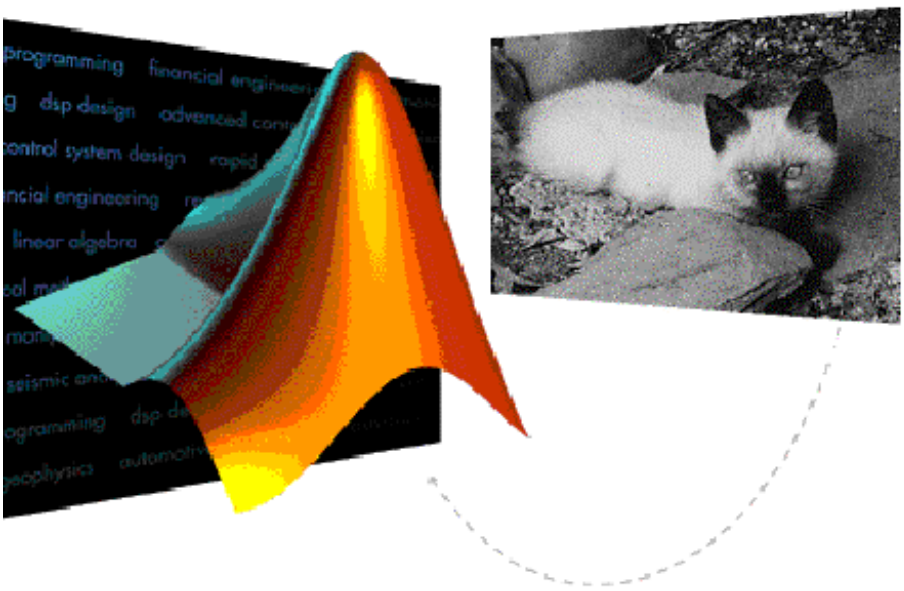
UL8 – PROJET MATLAB

TRAITEMENT D'IMAGES
CONVOLUTION ET RESTAURATION D'IMAGES

RIVIERE Nicolas

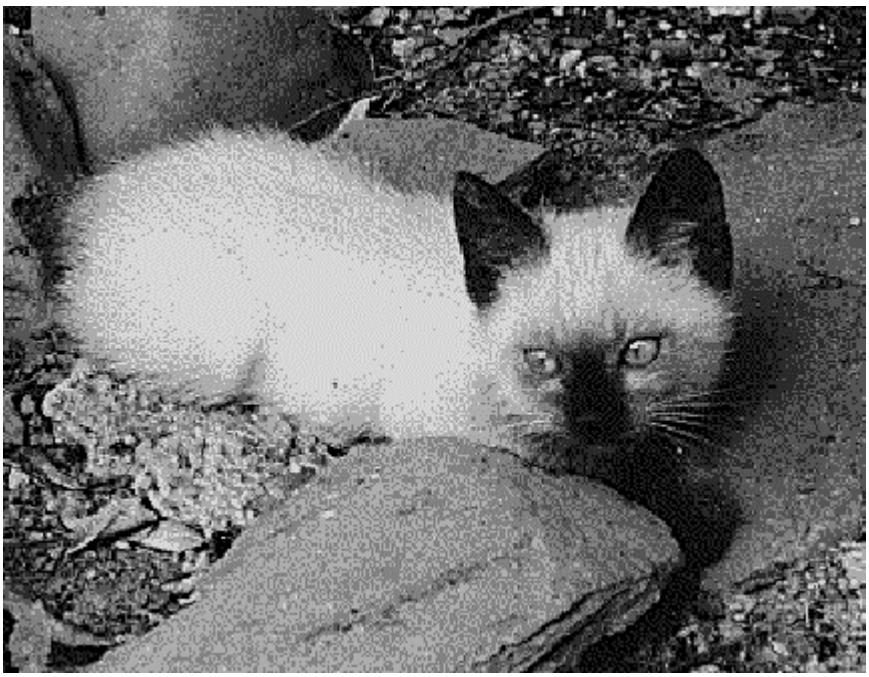


Université Paul Sabatier – Toulouse III - Licence de Physique - Juin 2000

SOMMAIRE

RAPPEL DU SUJET

A partir d'un signal à deux dimensions et considéré comme étant idéal, ce projet consiste à étudier les principes de convolutions numériques. Après avoir modélisé une chaîne d'acquisition, nous essaierons de reconstituer un signal correct, dépourvu de tout bruit parasite. Enfin, nous regarderons quelle méthode est à privilégier pour l'étude d'une image dégradée, celle d'un chat.



INTRODUCTION

Les objets auxquels s'intéressent notamment l'astrophysique, la télédétection spatiale ou la météorologie, ne sont, par nature, pas accessibles à l'expérience directe (pas de contrôle des paramètres externes). Ainsi l'imagerie constitue-t-elle l'unique moyen d'accéder aux paramètres physiques et géométriques des systèmes étudiés.

La distance géométrique qui sépare l'expérimentateur du système impose l'utilisation d'instruments; en effet la seule observation visuelle ne permet pas de collecter une quantité suffisante d'informations car l'œil humain - qui peut être considéré comme un instrument - a des performances limitées (fréquence de coupure incohérente, résolution angulaire, sensibilité, …).

La mise en place de tels dispositifs peut être décrite par la situation suivante :

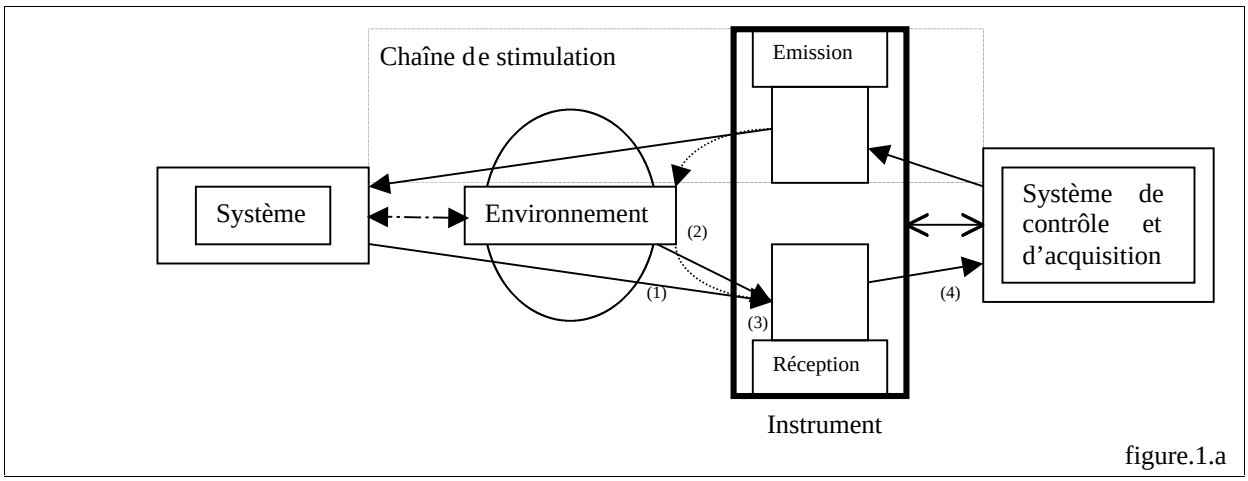


figure.1.a

(1) Informations sur le système modifiées éventuellement par l'environnement.
(2) Informations sur l'environnement = parasites et/ou bruit de fond.
(3) Fonction de transfert de l'appareil (défauts + réponse).
(4) Bruit d'électronique.

Lorsque la chaîne de stimulation (partie supérieure du diagramme) n'est pas présente, on aboutit au diagramme suivant, où le problème de l'interaction entre la stimulation et l'environnement ne se pose pas.

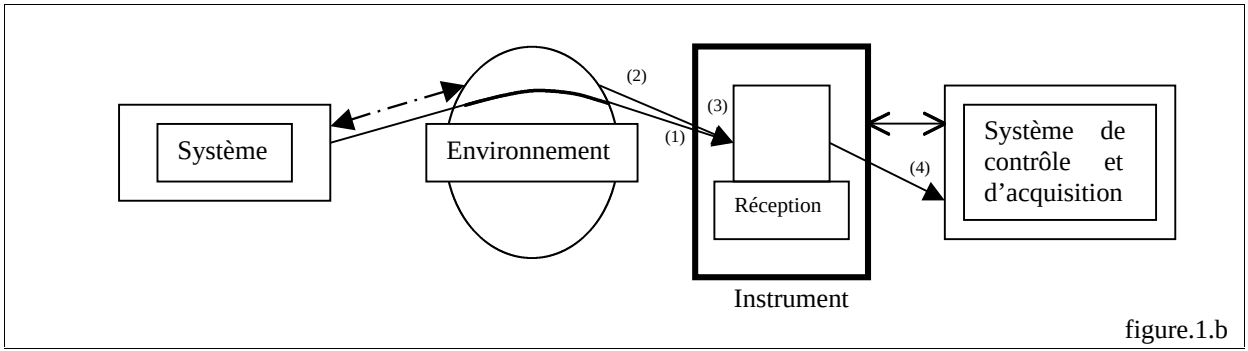


figure.1.b

Evidemment dans ce cas on n'a plus de contrôle sur l'environnement. Si on considère, dans un premier temps, que l'information à collecter est constituée de la réunion de celle relative au système et de celle relative à l'environnement, on doit alors s'affranchir des perturbations liées à l'instrument. Pour cela deux possibilités sont envisageables :

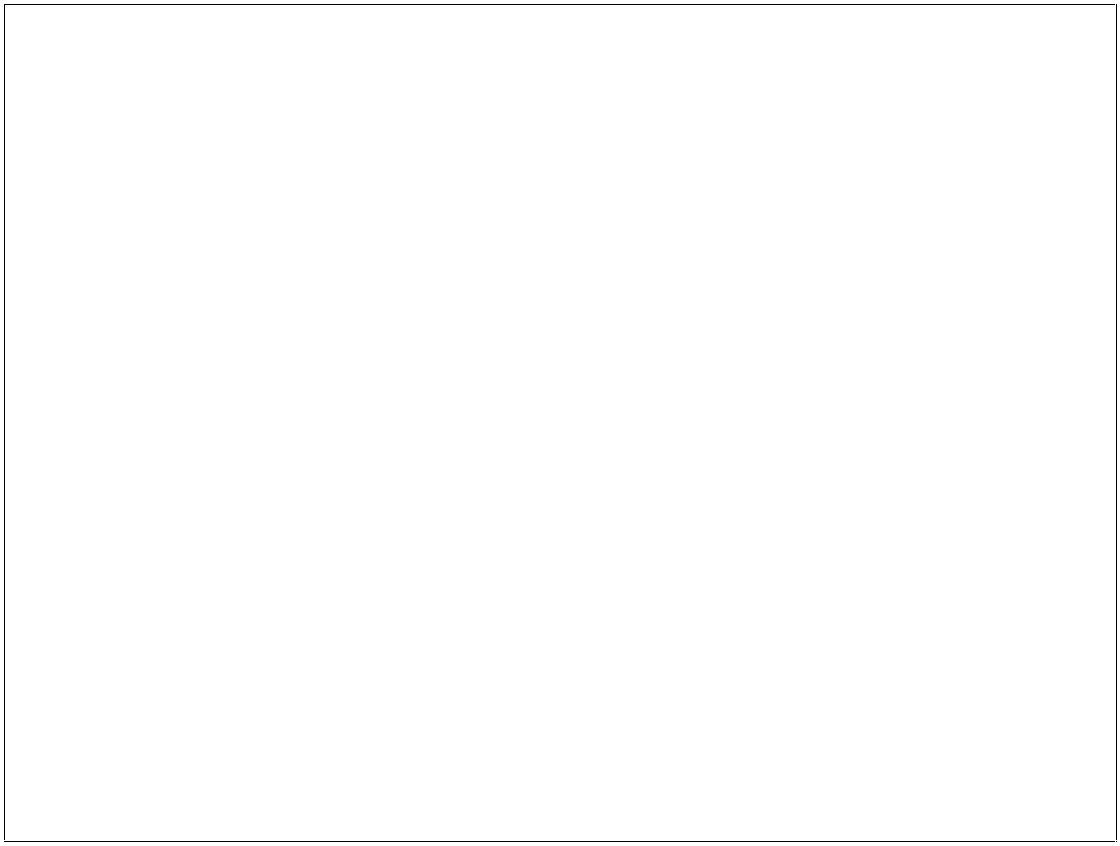
- soit améliorer les performances des instruments,
- soit recourir au traitement numérique.

Le premier cas induit nécessairement une augmentation du budget alloué à la réalisation. En revanche, la seconde solution est aisée à mettre en œuvre et permet de diminuer le coût de conception.

POSITION DU PROBLEME - DEFINITIONS

Pour illustrer les principales **transformations** que l'on peut opérer sur une image nous prendrons l'image d'un ***chat***.

Afin de simuler le résultat de l'acquisition de l'image par un instrument (**chaîne d'acquisition**), nous avons convolué l'image initiale *I* par une matrice représentant une **fonction de transfert** gaussienne, à la suite de quoi nous avons superposé un **bruit** (à **densité spectrale** constante), matrice générée par la fonction *rand* de MatLab :



Pour générer la matrice de flou *F*, nous avons construit une matrice carrée de ***rang p*** impair dont la valeur centrale ([(p+1)/2 ; (p+1)/2]) correspond à la valeur maximale de la gaussienne (=1) et dont l'extension, en **pixels**, est fixée par le paramètre ***sigma***. L'image dégradée est obtenue en appliquant la fonction *conv2(chat, flou, 'same')* [1].

[1] La condition *'same'* permet de conserver la taille originale de l'image après convolution.

Le ***bruit*** *B* a les mêmes dimensions que l'image ainsi, on peut additionner ces deux matrices pour simuler le résultat *R* de l'opération d'acquisition de l'image par un instrument sommaire.

Il convient maintenant de déterminer quelle procédure nous devons utiliser pour restituer l'image initiale en veillant à **minimiser** la perte d'information. En effet, on peut écrire $\boxed{R = I * F + B}$. En inversant cette relation, on peut (théoriquement) retrouver l'image initiale selon $I = (R - B) * F^{-1}$. Or *B* ne peut pas être strictement déterminé, ce qui complique le problème.

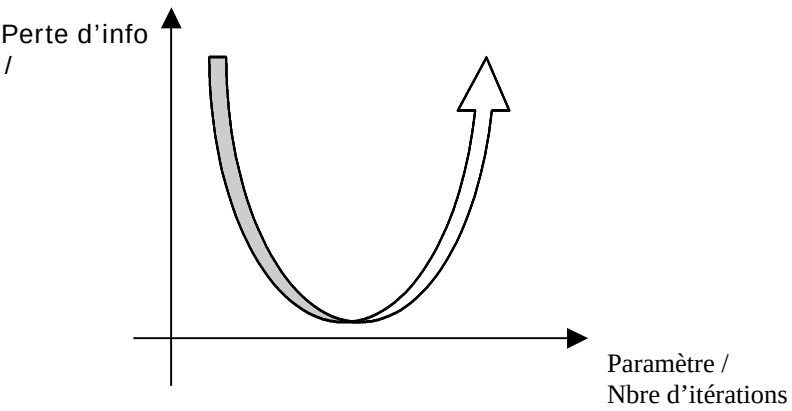
Cette première analyse nous conduit donc à rechercher un algorithme visant à estomper voire à extraire le bruit de l'image. Pour cela deux voies nous semblent intéressantes :

- les transformations ponctuelles (moyenne et médiane),
- la transformée de Fourier.

METHODES DE TRAITEMENT

o TRAITEMENT, CONSERVATION ET PERTE DE L'INFORMATION

Avant d'étudier plus en détails les transformations évoquées précédemment, il est important de remarquer que **tout traitement engendre nécessairement une perte d'information**. Pour l'illustrer, considérons une image à laquelle on a ajouté du bruit ; on peut montrer que la convolution de cette image par une gaussienne équivaut à un filtre passe-bas permettant ainsi de réduire le bruit souvent caractérisé par des hautes fréquences spatiales. La fréquence de coupure est fonction de la largeur de la gaussienne et, si celle-ci est trop étroite dans l'espace réciproque, on perd toute l'information. Le graphique suivant nous impose un « savant dosage » entre un nombre d'itérations trop faible ou trop important.



- Représentation qualitative de la perte d'information -

o TRANSFORMATIONS PONCTUELLES

MOYENNE

Si on considère un **environnement** 3x3, la moyenne est la valeur que l'on affecte au pixel central et qui est la moyenne (pondérée ou non) des valeurs des **niveaux** des pixels de l'environnement. Dans l'image qui nous intéresse, les niveaux se répartissent de 0 à 255 [2], 0 pour le *noir*, 255 pour le *blanc*.

La moyenne non pondérée se traduit par la convolution de l'image par la matrice $\frac{1}{9}\cdot\begin{pmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{pmatrix}$.

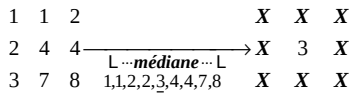
[2] 0 à 255 représente 256 niveaux *i.e.* 2^8 on pourra donc coder sur 8 bits.

- Chat original -

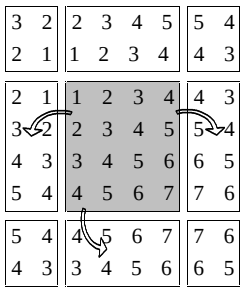
- Chat traité par la méthode Médiane -

MEDIANE

La méthode de la médiane quant à elle s'effectue de la manière suivante : on **classe** les valeurs de l'environnement **par ordre croissant** et on affecte au pixel central le niveau du pixel situé au milieu de la liste c'est à dire la **médiane au sens statistique** d'où le nom donné à cette transformation.



Remarque : De par son principe, la convolution numérique de matrices pose un problème aux « bords » de la matrice. Pour pallier cet inconvénient, on augmente la taille de l'image en appliquant un **miroir** sur les quatre directions. On **conserve** ainsi toute l'**information statistique** contrairement à la méthode qui consiste à agrandir l'image avec des « zéros ».



q TRANSFORMATION SPECIALE

Nous avons développé une troisième méthode qui s'appuie sur l'**autocorrélation** et sur les propriétés statistiques de la moyenne par rapport au bruit. En effet, l'autocorrélation augmente généralement le rapport signal sur bruit, ce qui est notre but recherché. Mais, nous voulons conserver l'information et la structure locale de l'image. Nous avons donc effectué, dans une première étape, une **autocorrélation locale** (typiquement, un environnement n x n). Le résultat ainsi obtenu (pavé n x n) peut, en terme de valeurs, être considéré comme une élévation au carré de l'intensité. Une question se pose à ce stade de la réflexion : doit-on affecter la valeur moyenne du résultat puis en prendre la racine carrée ou, doit-on prendre la racine carrée avant de réaliser la moyenne ? Dans le programme, nous avons choisi de réaliser la première solution.

Remarque : Si on ne prend pas le soin de prendre la racine carrée, on obtient une saturation de la dynamique de l'image (image trop sombre).

q REDUCTION DE BRUIT ET MOYENNE

Une solution peut être envisagée afin de conserver les propriétés locales de l'image : l'**autoconvolution locale**. Dans ce cas, on tient compte de la structure locale de l'image et on élimine le bruit qui n'a pas de corrélation entre les différents points et qui se trouve ainsi gommé par l'autoconvolution. En appliquant la méthode de la moyenne décrite ci-dessus, sur la zone calculée, on augmente l'efficacité du processus d'atténuation de bruit tout en limitant la perte d'information.
La combinaison de ces deux méthodes (moyenne et convolution) nous donne d'excellents résultats lorsqu'il s'agit de traiter un signal fortement bruité.

q REDUCTION DE BRUIT PAR ALGORITHME

Jusqu'à présent, nous réalisions une étude non itérative de l'image. L'algorithme de **Van-Cittert** permet de traiter le signal en appliquant un processus **itératif** qui atténue le bruit. Le problème lié à cette méthode consiste au fait que l'on ne sait pas exactement quand il faut stopper la récurrence. Un examen des résultats montre qu'il suffit de quelques itérations (une dizaine) pour converger vers une image plus nette. De plus, si l'on augmente le nombre de passages, on détériore l'image en faisant apparaître des artefacts de calculs. Il convient donc de maîtriser les paramètres avant de réaliser cet algorithme.
Le coefficient de relaxation $\boldsymbol{a}$ est en général assez difficile à mettre en œuvre mais, nous pouvons lui donner une valeur comprise entre 0 et 1. Pour l'algorithme de Van-Cittert, on aura $\boldsymbol{a}$ qui sera égal à 1 alors que pour celui de Jansson, le coefficient est proche de 0,1. Ce terme apparaît dans l'algorithme de Van-Cittert :

$$\boldsymbol{I_{k+1}}(x,y) = \boldsymbol{I_k}(x,y) + \boldsymbol{a}(x,y) \cdot \left[\boldsymbol{I_0}(x,y) - (\boldsymbol{I_k}(x,y) \otimes \boldsymbol{PSF})\right]$$

… avec $\boldsymbol{k}$ ordre de récurrence

$\boldsymbol{a}$ coefficient de relaxation

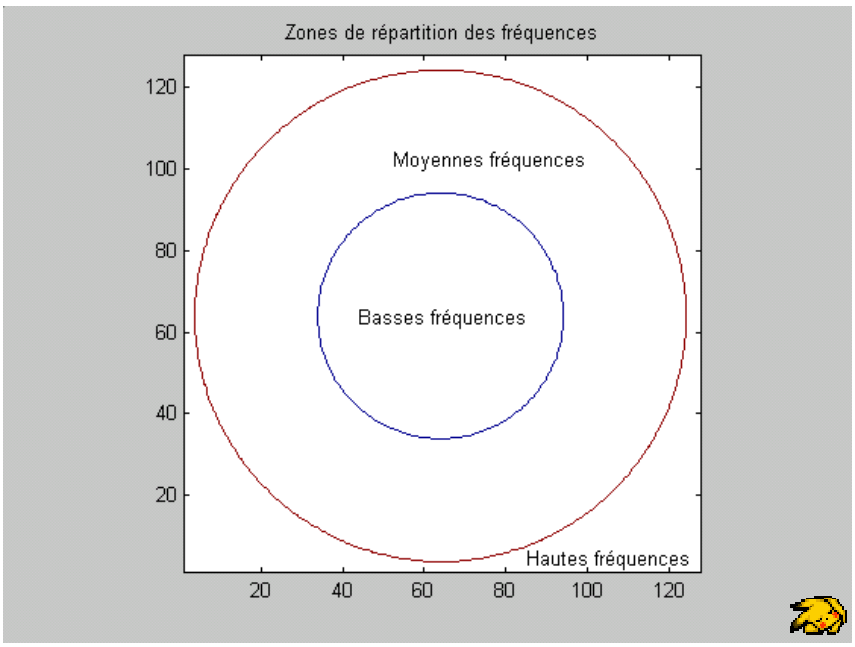
$\boldsymbol{I_0}$ Image à traiter

On voit qu'il est nécessaire de connaître la PSF. Il existe des algorithmes permettant de la déterminer mais, nous n'aborderons pas ce sujet dans le présent rapport. Nous partirons du principe qu'elle peut être déterminée expérimentalement.

o TRANSFORMEE DE FOURIER

La transformation de Fourier s'appuie sur la **décomposition** d'un signal en somme de fonctions périodiques (combinaisons linéaires des fonctions $e^{i\cdot\vec{k}\vec{x}}$ qui forment une base orthonormale de l'espace de Hilbert). Cette **représentation** contient **autant d'information** que le signal dans l'**espace direct** $\vec{x}$. Elle présente donc l'avantage de faire ressortir les structures périodiques ainsi que de révéler le bruit qui se traduit généralement par des hautes fréquences spatiales dans l'**espace réciproque** $\vec{k}$. De cette manière on peut diminuer le bruit en appliquant un filtre qui sera la plupart du temps un filtre passe-bas. Cependant, si on analyse les domaines de fréquences spatiales et leurs contenus informationnels, on s'aperçoit que les basses fréquences représentent à la limite le fond continu, les hautes fréquences - comme on l'a déjà évoqué – sont assimilables au bruit, quant aux fréquences moyennes on peut considérer qu'elles véhiculent les détails. On se trouve donc confronté au choix délicat de la fréquence de coupure du passe-bas, qui, si elle est trop basse, risque d'entraîner une trop grande perte d'information.

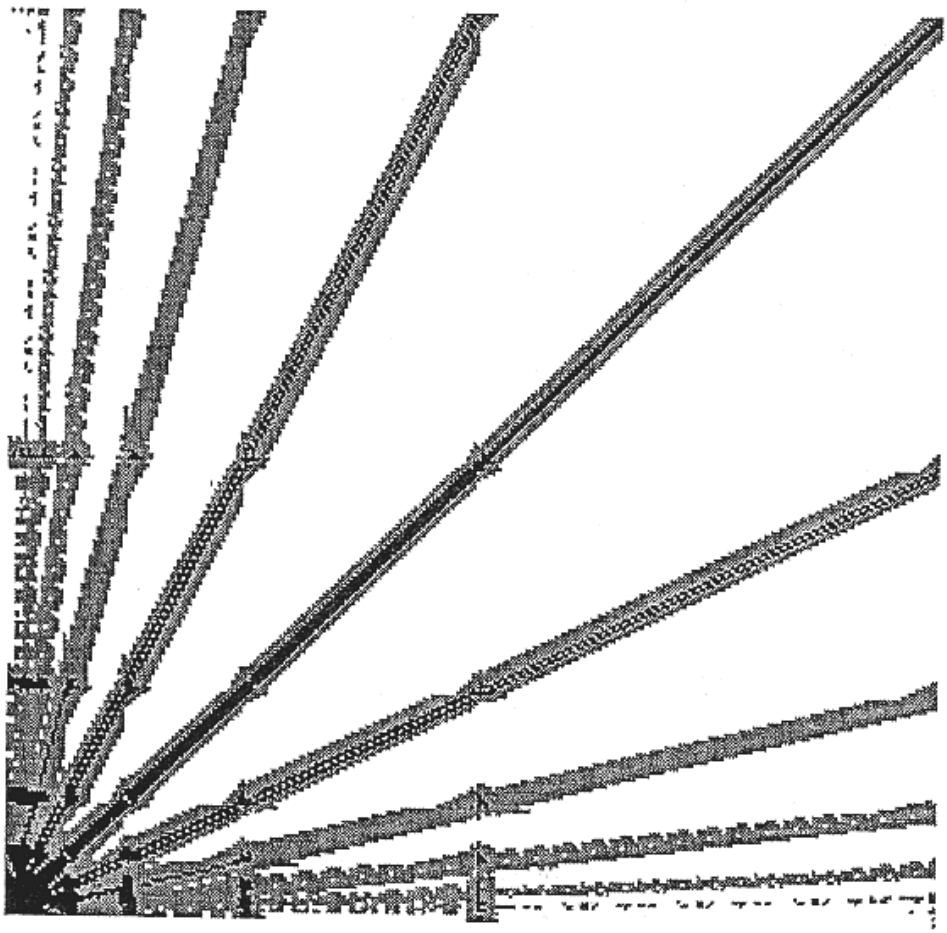
Le principal avantage de la T.F. est de transformer le produit de convolution $f * g$ en un simple produit des transformées de Fourier $\hat{f}\cdot\hat{g}$. Ainsi tous les traitements complexes de convolution par des filtres pourront être réalisés de manière simple, à condition de déterminer la **réponse caractéristique** du filtre, c'est à dire la T.F de la fonction de convolution qui définit le filtre.
En revanche, cette transformation étant globale, si on modifie un coefficient de la décomposition alors, cela se répercute sur tout le signal. Ce principal inconvénient nous pousse donc à rechercher une méthode aussi performante que la T.F. mais pouvant agir localement sur le signal.



o **TRANSFORMEE PAR ONDELETTES**

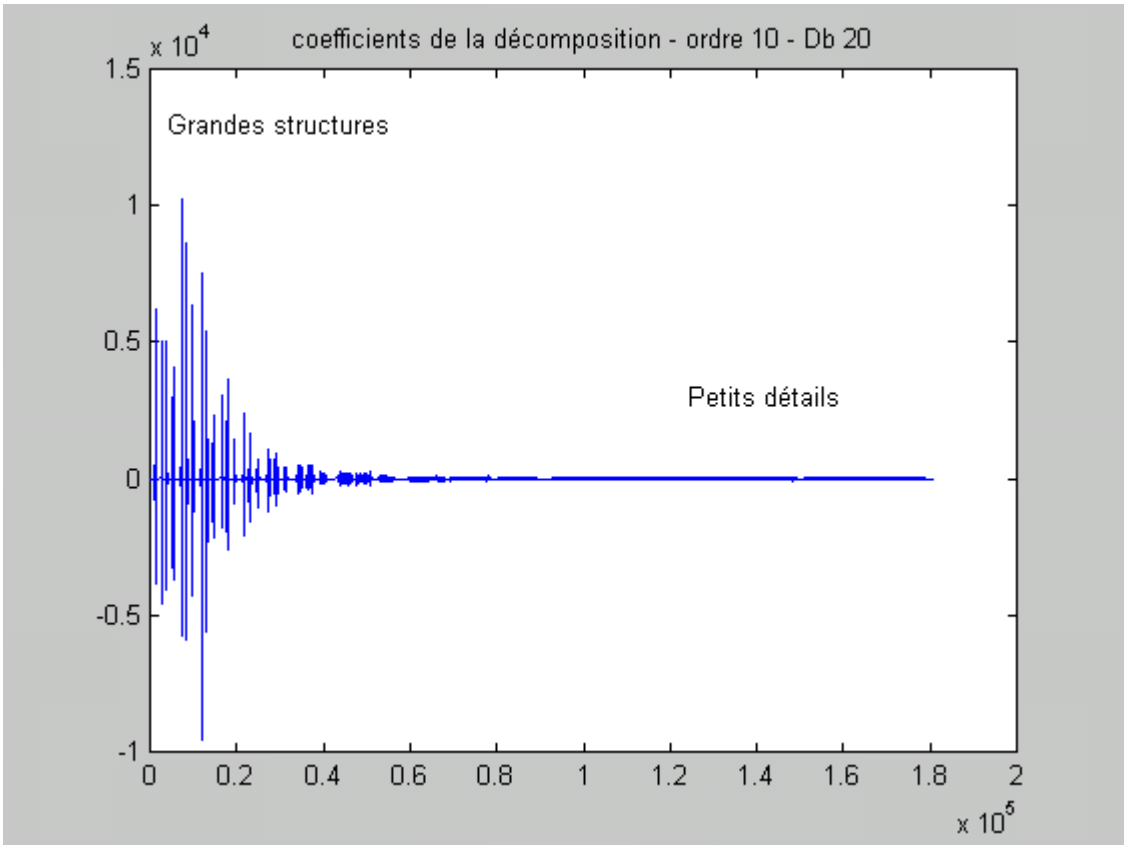
Cette transformation se distingue de la méthode précédente car elle privilégie une étude **locale** et non globale de l'image : la décomposition se fait, non plus dans l'espace des fonctions périodiques, mais sur une autre classe de fonctions telles que les ondelettes de Daubechies, Lemarie , *Coiflets*, *Symlets*…Cette technique combine les informations de position et de fréquence spatiale. Tout comme il existe la FFT, la DWT (Discrete Wavelet Transform) permet d'obtenir la transformée d'ondelettes d'ordre un, de l'image à l'aide d'un algorithme rapide.
On peut ainsi extraire les détails de l'image.

Le principe essentiel de cette technique réside dans la décomposition hiérarchique de l'image, c'est à dire que l'on peut visualiser les détails suivant leur direction (horizontal, vertical, diagonal) et suivant leur taille caractéristique. Ceci peut être réalisé à l'aide de la fonction *dwt2* incluse dans la librairie Wavelets de MatLab 5.3.



Transformée par ondelettes d'une matrice 256 x 256

Dans la suite du problème, nous préférerons utiliser la commande *wavedec2* qui calcule la décomposition en ondelettes d'un signal en deux dimensions et qui permet d'accéder aux coefficients de la transformée afin de les modifier. En effet, en considérant les détails les plus fins comme étant du bruit, on peut raisonnablement espérer nettoyer l'image en les atténuant, voire en les supprimant.

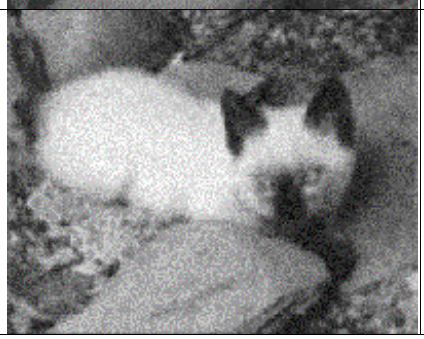


Cette méthode a retenu notre attention par rapport aux précédentes car elle possède certains avantages non négligeables. La rapidité de calcul, même pour des ordres de décomposition assez élevés, et la bonne discernabilité des structures en font une technique généralement très utilisée dans les domaines de retraitement d'images (astronomie, stockage de données compressées, …).

COMPARAISON DES RESULTATS

Nous pouvons regrouper l'ensemble des résultats dans le tableau ci-dessous :

Nom	Image obtenue	Commentaires
Image initiale		Aucune déformation de l'image
Image à étudier		Flou + Bruit
Moyenne & Médiane		Rapidité : *très lent* Aspect visuel : *bon rendu* Difficultés : *effet miroir aux bords* Paramètres par défaut : *matrice 3 x 3* Parfait pour image fortement bruitée Peu de différence entre moyenne et médiane
« Autocorrélation »		Rapidité : *très lent* Aspect visuel : *bon rendu* Difficultés : *effet miroir aux bords* Paramètres par défaut : *matrice 3 x 3* Garde la structure principale

Transformée de Fourier		Rapidité : *lent* Aspect visuel : *rendu moyen* Difficultés : - Paramètres par défaut : - Traitement global de l'image
Transformée par ondelettes		Rapidité : *très rapide* Aspect visuel : *très bon rendu* Difficultés : *Déterminer l'ordre* Paramètres par défaut : *ordre 11* Traitement local de l'image
Algorithme de Van-Cittert		Rapidité : *lent* Aspect visuel : *bon rendu* Difficultés : *Nombre d'itérations* Paramètres par défaut : *10 itérations* Parfait pour image fortement bruitée

CONCLUSION

Lors de la réalisation de cet exposé et de la conception des programmes, nous avons rencontré des difficultés liées aux choix des algorithmes. Certaines méthodes sont préférables à d'autres mais, il n'existe pas de traitement universel : à chaque image son traitement. De plus, nous avons été confrontés à la détermination des bons paramètres de réglage et à l'affichage correct de l'image finale. Le réglage de la dynamique représente un point important du traitement de signaux à deux dimensions même si nous ne nous sommes pas trop étendus sur ce sujet.

Toutes les méthodes testées précédemment ont donc plus ou moins d'effets sur le signal. Néanmoins, on peut retenir la technique s'appuyant sur les ondelettes. Cette dernière est d'ores et déjà appliquée pour la compression de données (ex. le F.B.I.) car, en ôtant certains coefficients faibles, on réduit la taille mémoire de l'image et on conserve l'information essentielle (*i.e.* les structures principales).
La moyenne ou la médiane retirent également le bruit de l'image mais l'algorithme ne semble pas adapté à un logiciel tel que MatLab. En effet, la longueur des opérations et la contrainte des boucles imposent un temps d'attente non négligeable.
Les méthodes consistant à traiter l'image par récurrence (cf. algorithme de Van-Cittert) peuvent être intéressantes dès que l'on tente de détecter des objets ponctuels comme des étoiles dans un ciel profond. Pour une image « classique » telle que celle du chat, nous n'apportons que peu d'intérêt à ces techniques. Dans le cadre de l'étude proposée, l'information primordiale doit être dégagée du signal bruité.

Il existe d'autres méthodes qui sont très largement utilisées dans le monde des « chirurgiens » de l'images. Notre but n'étant pas d'écrire un catalogue des différentes techniques mais de s'initier aux notions et aux outils développés pour le traitement d'image, nous avons délibérément laissé de côté les problèmes de seuillage, de reconnaissance de formes, de squelettisation, …

BIBLIOGRAPHIE

Publications

« Identification of structures from galaxy counts : use of the wavelet transform » E. SLEZAK – A. BIJAOUI – Astron. Astrophys. 227, 301-316 (1990)

« Image reconstruction by the wavelet transform applied to aperture synthesis » J. L. STARCK – A. BIJAOUI – Astron. Astrophys. 283, 349-360 (1994)

« Image restoration with noise suppression using a multiresolution support » F. MURTAGH - J. L. STARCK – A. BIJAOUI – Astron. Astrophys. Suppl. Ser. 112, 179-189 (1995)

Livres

« Introduction aux techniques de traitement d'images » André MARION – Editions EYROLLES

« Précis d'analyse d'images » Michel COSTER - J. L. CHERMANT - Editions Presses du CNRS

« Le guide pratique de l'astronomie CCD » Patrick MARTINEZ – Alain KLOTZ – Editions ADAGIO

« Optique – Fondements et applications » José-Philippe PEREZ – Editions MASSON

« Traitement d'images & architectures parallèles » Richard DAPOIGNY – Editions ADDISON-WESLEY

« Analyse d'images : filtrage et segmentation » J. P. COCQUEREZ – S. PHILIPP – Editions MASSON

Sites Internet

Filtrage récursif adaptatif

http://www-sim.int-evry.fr/~regalia/iirvf.html

Image processing toolbox – Traitement d'images

http://www.ssg.fr/tmw/produits/toolboxes/image.html

Wavelet toolbox

http://www.ssg.fr/tmw/produits/toolboxes/wavelet.html

ANNEXES

Publications

« Identification of structures from galaxy counts : use of the wavelet transform » E. SLEZAK – A. BIJAOUI – Astron. Astrophys. 227, 301-316 (1990)

« Image reconstruction by the wavelet transform applied to aperture synthesis » J. L. STARCK – A. BIJAOUI – Astron. Astrophys. 283, 349-360 (1994)

« Image restoration with noise suppression using a multiresolution support » F. MURTAGH - J. L. STARCK – A. BIJAOUI – Astron. Astrophys. Suppl. Ser. 112, 179-189 (1995)

Scripts des programmes

Prog.m
Ch_acqui.m
Dynamique.m
Convol.m
Reconst.m
Moyenne.m
Mediane.m
Cittert.m
Moyconv.m
Ondelet.m
…

Prog.m
Programme principal à exécuter pour visualiser le projet.

```
%
%  Prog. Principal commenté
%
%  Licence de Physique 2000 - UL8
%  Projet MatLab : Traitement d'images
%
%  RIVIERE Nicolas & DELAGNES Jean-Christophe
%

%  Initialisation
close all; clear all; clc;
global t;

%  Présentation du programme
dos('\affich\dzlrun.exe \affich\cd01.mcl');
Presentation;

%  Modélisation de la chaîne d'acquisition
dos('\affich\dzlrun.exe \affich\cd02.mcl');
Ch_acqui;

%  Représentation des images
subplot(2,2,1);
image(chat);
colormap(lut);
axis off;
title('Chat à photographier');
subplot(2,2,2);
image(chat_flou);
colormap(lut_flou);
axis off;
title('Chaîne d''acquisition 1');
figure;
image(chat);
colormap(lut);
axis off;
title('Chat à photographier');
figure;
image(chat_flou);
colormap(lut_flou);
axis off;
title('Chaîne d''acquisition');

%  Reconstitution de l''image
dos('\affich\dzlrun.exe \affich\cd03.mcl');
reconst;
prog_fin;
```

Ch_acqui.m
Ce programme simule une chaîne d'acquisition en rendant flou l'image et en appliquant du bruit sur l'original.

```
%
%  Sous Prog. Modélisation de la chaîne d'acquisition
%
%  Licence de Physique 2000 - UL8
%  Projet MatLab : Traitement d'images
%
%  RIVIERE Nicolas & DELAGNES Jean-Christophe
%

global t;

disp ' '
disp '  Modélisation de la chaîne d''acquisition'
disp ' '
disp '     Chargement de l''image'

load chat;
taille   =  size(chat);
tx       =  taille(1);
ty       =  taille(2);


disp ' '
disp '     Réalisation de la PSF'
chat_flou   =  convol(chat);

disp ' '
disp '     Définition du bruitage de l''image'
pause(1);

prompt   =  {'Entrez la valeur du paramètre de signal sur bruit ie SNR'};
def={'2'};
dlgTitle =  'Modélisation de la chaîne d''acquisition';
lineNo=2;
dlgresult=  inputdlg(prompt,dlgTitle,lineNo,def);
pause(0.5);
dlgresult(1);
SNR      =  str2num(cat(1,ans{:}));
if isempty(SNR)
   SNR   =  2;
end;
pause(0.5);
bruit    =  (255/SNR)*rand(tx,ty);

disp ' '
disp '     Modèle restitué par la chaîne d''acquisition'
chat_flou   =  chat_flou + bruit;

%  Passage à une image codée en 8 bits
chat_flou   =  (255/max(max(chat_flou))) * chat_flou;

%  Correction de la dynamique de l'image
lut      =  dynamique(chat);
lut_flou =  dynamique(chat_flou);
```

Dynamique.m
Fonction permettant de modifier la dynamique de l'image.

```
%
%  Fonction Correction de la dynamique d'une image
%
%  Licence de Physique 2000 - UL8
%  Projet MatLab : Traitement d'images
%
%  RIVIERE Nicolas & DELAGNES Jean-Christophe
%

function  dynamique  =  lut(nom);

%  Initialisations
format long;
hh=3.0;

%  Définiton des seuils de visualisation
sh=max(max(nom));
sb=min(min(nom))+1;

%  Changement de table de couleurs
if (sb<sh)
   haut=1;
   bas=0;
else
   haut=0;
   bas=1;
   a=sh;sh=sb;sb=a;
end

%  Correction de la dynamique
sb    =  1;
sh    =  255;
dyna  =  255;
intensite(1:sb)=linspace(bas,bas,sb);
intensite(sb:sh)=linspace(bas,haut,sh-sb+1);
intensite(sh:dyna)=linspace(haut,haut,dyna-sh+1);
lut(:,1)=intensite';
lut(:,2)=intensite';
lut(:,3)=intensite';

%  Retour au prog.
dynamique   =  lut;
return
```

Convol.m
Fonction permettant de modifier la dynamique de l'image.

```
%
%  Fonction - Modélisation de la chaîne d'acquisition - Convolution
%
%  Licence de Physique 2000 - UL8
%  Projet MatLab : Traitement d'images
%
%  RIVIERE Nicolas & DELAGNES Jean-Christophe
%

function  convol  =  ti(nom);
% avec nom   = nom de la matrice image à traiter

global t;

% Paramétrage
prompt   =  {'Entrez la taille de la matrice de convolution. Si la taille de
la matrice carrée est paire, elle sera arrondie à l''entier impair
supérieur.'};
def={'3'};
dlgTitle =  'Modélisation de la chaîne d''acquisition - Convolution';
lineNo=2;
dlgresult=  inputdlg(prompt,dlgTitle,lineNo,def);
pause(0.5);
dlgresult(1);
t_trou   =  str2num(cat(1,ans{:}));
if isempty(t_trou)
   t_trou   =  3;
end;

% Vérification de la parité
if rem(t_trou,2)==1
   t_trou = t_trou+1;
end;

% Réalisation de l'objet
m_trou   =  ones(t_trou,t_trou);
for i=1:t_trou
   for j=1:t_trou
      if ~( ((i-(t_trou+1)/2)^2+(j-(t_trou+1)/2)^2)<(((t_trou+1)/2)^2) )
         m_trou(i,j)=0;
      end;
   end;
end;

% Convolution
t        =  conv2(m_trou,m_trou);
ti       =  conv2(nom,t,'same');

% Régulation des couleurs
ti       =  255*ti/max(max(ti));

% Retour des valeurs
convol   =  ti;
return
```

Reconst.m
Pour reconstituer l'image initiale, ce sous programme présente sous forme de menu l'ensemble des méthodes possibles.

```
%
%  Sous Prog. Reconstitution de l'image
%
%  Licence de Physique 2000 - UL8
%  Projet MatLab : Traitement d'images
%
%  RIVIERE Nicolas & DELAGNES Jean-Christophe
%


% Parmamétrage des variables
fin   =  0;


% Boucle permettant de tester chaque méthode
while fin ~= 1
clc;
pause(0.5);
disp ' '
disp '  Reconstitution de l''image'
pause(1);

choix =  questdlg('Quel traitement souhaitez-vous appliquer à l''image obtenue
après modélisation de la chaîne d''acquisition ?', ...
                        'Convolution et restauration d''images', ...
                        'Moyenne /
Médiane','Transformées','Algorithmes','Fin');

switch choix,
   case 'Moyenne / Médiane'
      choix =  questdlg('Sélectionnez la méthode a utiliser.', ...
                        'Convolution et restauration d''images', ...
                        'Moyenne','Médiane','Fin');
   case 'Transformées'
      choix =  questdlg('Sélectionnez la méthode à utiliser.', ...
                        'Convolution et restauration d''images', ...
                        'Ondelettes','Espace de
Fourier','"Autoconvolution"','Fin');
   case 'Algorithmes'
      choix =  questdlg('Sélectionnez l''algorithme à appliquer.', ...
                        'Convolution et restauration d''images', ...
                        'Van Cittert','Fin');
end; % switch


close all; pause(0.5);

switch choix,
     case 'Moyenne',
         prompt   =  {'Entrez la taille de la matrice carrée'};
         def      =  {'3'};
         dlgTitle =  'Traitement de l''image - Moyenne';
         lineNo   =  2;
```

```
        dlgresult=  inputdlg(prompt,dlgTitle,lineNo,def);
        dlgresult(1);
        t        =  str2num(cat(1,ans{:}));
        if isempty(t)
              t  =  3;
        end; %if
        im_traitee  =  moyenne(chat_flou,t);

    case 'Médiane',
        prompt   =  {'Entrez la taille de la matrice carrée'};
        def      =  {'3'};
        dlgTitle =  'Traitement de l''image - Médiane';
        lineNo   =  2;
        dlgresult=  inputdlg(prompt,dlgTitle,lineNo,def);
        dlgresult(1);
        t        =  str2num(cat(1,ans{:}));
        if isempty(t)
              t  =  3;
        end; %if
        im_traitee  =  mediane(chat_flou,t);

    case '"Autoconvolution"',
        prompt   =  {'Entrez la taille de la matrice carrée'};
        def      =  {'3'};
        dlgTitle =  'Traitement de l''image - Moyenne d''autoconvolution';
        lineNo   =  2;
        dlgresult=  inputdlg(prompt,dlgTitle,lineNo,def);
        dlgresult(1);
        t        =  str2num(cat(1,ans{:}));
        if isempty(t)
              t  =  3;
        end; %if
        im_traitee  =  moyconv(chat_flou,t);

     case 'Ondelettes',
        im_traitee  =  ondelet(chat_flou);

     case 'Espace de Fourier',
        im_traitee  =  fourier(chat_flou);

     case 'Van Cittert',
        im_traitee  =  cittert(chat_flou);

     case 'Fin',
        fin   =  1;

       end % switch

% Affichage du résultat
if fin==0

disp ' '
disp '          Résultats obtenus après traitement de l''image'
pause(1);
fin2  =  0;
close all;
figure;
```

```
image(chat);
colormap(dynamique(chat));
axis off;
title('Chat initial');
figure;
image(chat_flou);
colormap(dynamique(chat_flou));
axis off;
title('Chat flou');
figure;
image(im_traitee);
colormap(dynamique(im_traitee));
axis off;
title(['Image traitée par la méthode ',choix]);
figure;
subplot(2,2,1);
image(chat); colormap(dynamique(chat)); axis off; title( 'Chat initial');
subplot(2,2,2);
image(chat_flou); colormap(dynamique(chat_flou)); axis off; title( 'Chat
flou');
subplot(2,2,3);
image(im_traitee); colormap(dynamique(im_traitee)); axis off; title( 'Chat
traité');


% Affichage par menu des différentes images
while fin2 ~= 1

   choix =  menu('Affichage de','Chat Initial','Chat flou','Chat
restitué','Comparaison','Retour menu méthodes');
   switch choix,
   case 1,
      figure(1);
   case 2,
      figure(2);
   case 3,
      figure(3);
   case 4,
      figure(4);
   case 5,
      fin2  =  1;
   end; % switch

end; % while affichage
end; %if Fin==0

end; %while méthodes

close all;
```

Moyenne.m
Fonction retournant l'image traitée par la méthode de moyenne.

```
%
%  Fonction - Création de l'image par la méthode Moyenne
%
%  Licence de Physique 2000 - UL8
%  Projet MatLab : Traitement d'images
%
%  RIVIERE Nicolas & DELAGNES Jean-Christophe
%

function  moyenne =  im_moyen(nom,t_mat);

% avec nom   = nom de la matrice à traiter
% et   t_mat = taille de la matrice carrée de traitement

% Initialisation
disp ' '
disp '          Méthode de la moyenne'
pause(0.3);
t        =  size(nom);
delta    =  (t_mat+1)/2;

% Création d'une matrice plus grande pour conditions aux bords : Effet miroir
nom      =  cat( 2, fliplr(nom(:,1:delta)) , nom , fliplr(nom(:,t(2)-
delta+1:t(2))) );
nom      =  cat( 1, flipud(nom(1:delta,:)) , nom , flipud(nom(t(1)-
delta+1:t(1),:)) );

figure;
im_moyen = nom;
axis off;
colormap(gray(255));
for x=delta:t(1)-delta+1

image(im_moyen);
text(-10,-10,[int2str(round(x*100/(t(1)-delta))),' %']);
pause(0.01);
for y=delta:t(2)-delta+1
      tab   =  nom( x:x+t_mat-1 , y:y+t_mat-1 );
      vect  =  0;
      for i=1:t_mat
      vect  =  [ vect, tab(i,:) ];
      end; % for i
      im_moyen(x+delta,y+delta)  =  mean(vect);

   end; %for y
end; %for x

% Extraction de l'image utile
im_moyen =  im_moyen(delta:t(1)-delta+1,delta:t(2)-delta+1);

% Retour des valeurs
moyenne  =  im_moyen;
return
```

Mediane.m
Fonction retournant l'image traitée par la méthode de la médiane.

```
%
%  Fonction - Création de l'image par la méthode Médiane
%
%  Licence de Physique 2000 - UL8
%  Projet MatLab : Traitement d'images
%
%  RIVIERE Nicolas & DELAGNES Jean-Christophe
%

function  mediane =  im_median(nom,t_mat);
% avec nom   = nom de la matrice à traiter
% et   t_mat = taille de la matrice carrée de traitement

disp ' '
disp '          Méthode de la médiane'

% Initialisation
pause(0.3);
t        =  size(nom);
delta    =  (t_mat+1)/2-1;

% Création d'une matrice plus grande pour conditions aux bords : effet miroir
nom      =  cat( 2, fliplr(nom(:,1:delta)) , nom , fliplr(nom(:,t(2)-
delta+1:t(2))) );
nom      =  cat( 1, flipud(nom(1:delta,:)) , nom , flipud(nom(t(1)-
delta+1:t(1),:)) );

figure;
im_median = nom;
axis off;
colormap(gray(255));
for x=delta:t(1)-delta+1

image(im_median);
text(-10,-10,[int2str(round((x-delta)*100/(t(1)))),' %']);
pause(0.01);
for y=delta:t(2)-delta+1
      tab   =  nom( x:x+t_mat-1 , y:y+t_mat-1 );
      vect  =  0;
      for i=1:t_mat
      vect  =  [ vect, tab(i,:) ];
      end; % for i
      im_median(x+delta,y+delta) =  median(vect);

   end; %for y
end; %for x

% Extraction de l'image utile
im_median   =  im_median(delta:t(1)-delta+1,delta:t(2)-delta+1);

% Retour des valeurs
mediane  =  im_median;
return
```

Cittert.m
Fonction retournant l'image traitée par l'algorithme de Van-Cittert.

```
%
%  Fonction - Traitement de l'image - Méthode Van-Cittert
%
%  Licence de Physique 2000 - UL8
%  Projet MatLab : Traitement d'images
%
%  RIVIERE Nicolas & DELAGNES Jean-Christophe
%

function  cittert =  res(nom);
% avec nom   = nom de la matrice image à traiter

disp ' '
disp '          Méthode Van Cittert'
pause(0.3);

% Définition des paramètres
global t;

prompt   =  {'A quel ordre souhaitez-vous développer les calculs ?','Entrez le
coefficient de relaxation :'};
def={'10','5'};
dlgTitle    =  'Traitement de l''image - Van Cittert';
lineNo=2;
dlgresult=  inputdlg(prompt,dlgTitle,lineNo,def);
pause(0.5);
dlgresult(1);
ordre =  str2num(cat(1,ans{:}));
if isempty(ordre)
   ordre =  10;
end;
dlgresult(2);
a  =  str2num(cat(1,ans{:}));
if isempty(ordre)
   a  =  5;
end;

% Calculs
pause(0.1);
im_traitee  =  nom;

for k=1:ordre
   im_traitee  =  im_traitee  +  a .* ( nom - conv2(im_traitee,t,'same') );
   im_traitee  =  255 * abs(im_traitee)/max(max(abs(im_traitee)));
   image(im_traitee); colormap(gray(255));
   title(['Ordre = ',int2str(k)]);
   pause(.1);
end;

% Prépondérence de certains "traits" de l'image
res   =  255*(nom+im_traitee)/(max(max(nom+im_traitee)));
% Retour des valeurs
cittert  =  res;
return
```

Moyconv.m
Fonction retournant l'image traitée par la méthode d'autocorrélation et de moyenne.

```
%
%  Fonction - Restitution de l'image - Moyenne de l'autoconvolution
%
%  Licence de Physique 2000 - UL8
%  Projet MatLab : Traitement d'images
%
%  RIVIERE Nicolas & DELAGNES Jean-Christophe
%

function  moyconv =  ti(nom,t_mat);

% avec nom   = nom de la matrice image à traiter

global t;
disp ' '
disp '          Moyenne de l''autoconvolution'
pause(0.3);

% Paramétrages
delta    =  (t_mat+1)/2;
taille   =  size(nom);

% Création d'une matrice plus grande pour effets de bords
nom      =  cat( 2, fliplr(nom(:,1:delta)) , nom , fliplr(nom(:,taille(2)-
delta+1:taille(2))) );
nom      =  cat( 1, flipud(nom(1:delta,:)) , nom , flipud(nom(taille(1)-
delta+1:taille(1),:)) );


% "Autocorrélatio"
for k=delta:taille(1)-delta+1
   for l=delta:taille(2)-delta+1
      a        =  nom((k-1):(k+1),(l-1):(l+1));
      b        =  conv2(a,a,'same');
      ti(k,l)  =  mean(mean(b));
   end;
   end;


% Régulation des couleurs
ti    =  255*sqrt(ti(delta:taille(1)-delta+1,delta:taille(2)-
delta+1)/max(max(ti)));

% Retour des valeurs
moyconv  =  ti;
return
```

Ondelet.m
Fonction retournant l'image traitée par la technique des ondelettes. Ce script fait appel à la librairie Wavelet incluse dans MatLab.

```
%
%  Fonction - Traitement de l'image - Méthode des ondelettes
%
%  Licence de Physique 2000 - UL8
%  Projet MatLab : Traitement d'images
%
%  RIVIERE Nicolas & DELAGNES Jean-Christophe
%

function  ondelet =  res(nom);

% avec nom   = nom de la matrice image à traiter

disp ' '
disp '          Méthode par ondelettes (wavelets)'
pause(0.3);

% Définition des paramètres
ok =  0;
while ok ~= 1
choix =  questdlg('Quelle méthode par ondelettes souhaitez-vous appliquer ?',
...
                        'Ondelettes - Filtres', ...
                        'Daubechies','Coiflets','Symlets','Fin');

pause(0.5);
switch choix,
      case 'Daubechies',
        fil    =  'db';
        bornes =  ' [1..45] :';
        ok     =  1;
     case 'Coiflets',
        fil    =  'coif';
        bornes =  ' [1..5] :';
        ok     =  1;
      case 'Symlets',
        fil    =  'sym';
        bornes =  ' [1..45] :';
        ok     =  1;
      case 'Fin',
end; % case
end; %while
pause(0.5);
prompt      =  {'Entrez l''ordre de décomposition :',['Entrez le numéro du
filtre de ',choix,bornes]};
def         =  {'3','2'};
dlgTitle    =  'Traitement de l''image - Ondelettes';
lineNo      =  2;
dlgresult   =  inputdlg(prompt,dlgTitle,lineNo,def);
dlgresult(1);
ordre       =  str2num(cat(1,ans{:}));
dlgresult(2);
num         =  cat(1,ans{:});
```

```
if isempty(ordre)
      ordre =  3;
end; %if
if isempty(num)
      num   =  '2';
end; %if
pause(0.1);


% Décomposition en ondelettes
[w_nom,s_nom] = wavedec2(nom,ordre,[fil,num]);

% Affichage et lecture graphique
pause(0.1);
figure;
plot(w_nom);
title('Sélectionnez la zone de coupure (axe vertical)');
coord    =  ginput(1);

prompt      =  {'Entrez le coefficient permettant de traiter les valeurs
supérieures à la coupure [0..1] :'};
def         =  {'0'};
dlgTitle    =  'Traitement de l''image - Ondelettes';
lineNo      =  2;
dlgresult   =  inputdlg(prompt,dlgTitle,lineNo,def);
dlgresult(1);
coef        =  str2num(cat(1,ans{:}));
if isempty(coef)
      coef  =  0;
end; %if
title('Traitement en cours');
pause(0.1);

w_nom(round(coord(1)):length(w_nom))   =
   w_nom(round(coord(1)):length(w_nom)).*coef;



% Décomposition Inverse ie Recomposition
res      =  waverec2(w_nom,s_nom,[fil,num]);

% Retour des valeurs
ondelet  =  res;
return
```

D'autres programmes sont présents sur le CD-Rom fourni avec le présent rapport. Leurs scripts ne nous semblent pas être intéressants pour être développés ici.