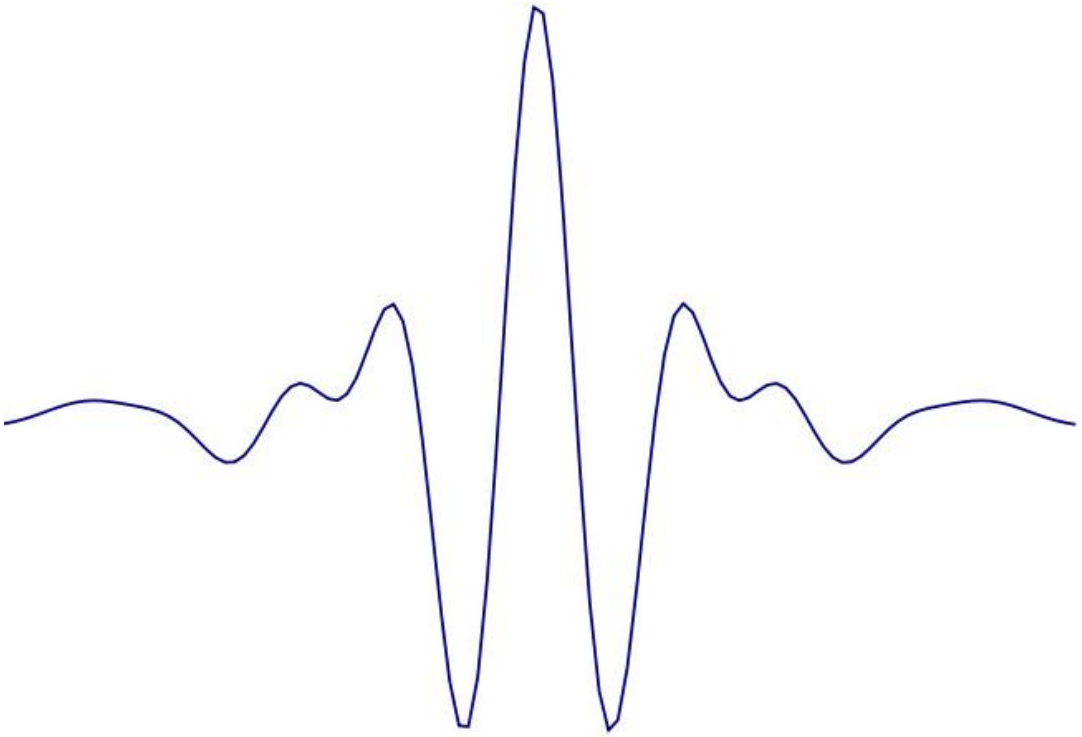
BRALET Antoine 4ETI
METEYER Paul

Travaux Pratiques Matlab

TP Ondelettes 2D

Débruitage, classification



Année 2018/2019

1. Débruitage d'une image : ondelettes orthogonales

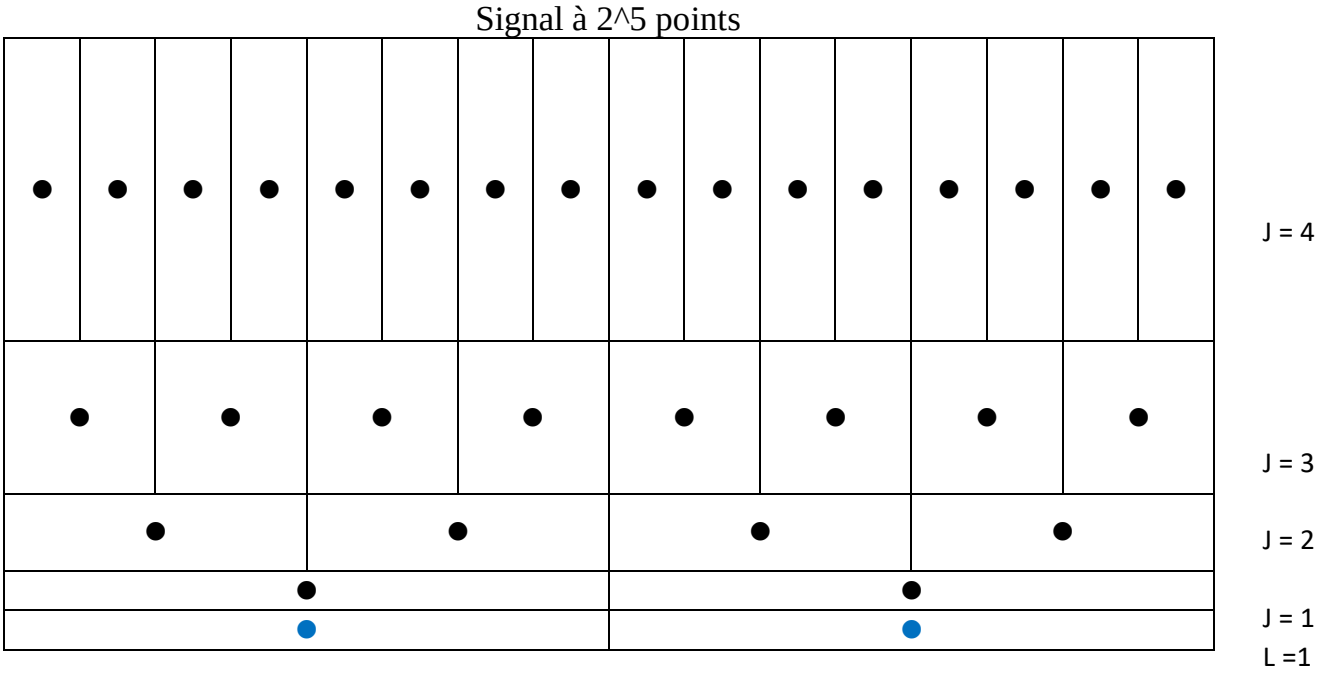
1.1 Analyse multi-résolution : décomposition/reconstruction

Le but de cette partie est d'effectuer la décomposition d'une image et de la reconstruire à partir d'une ondelette dite « Coiflet » qui a la particularité d'avoir N moments nuls.

Premièrement, avant d'appliquer la décomposition en ondelettes, il faut tout d'abord les visualiser. Ainsi pour créer l'ondelette Coiflet à trois moments nuls il faut tout d'abord calculer les coefficients du filtre en quadrature avec le code suivant :

```
%% Analyse multi-resolution
CoifQMF = MakeONFilter('Coiflet',3);
figure(1)
subplot(121)
plot(CoifQMF)
title('Coefficients du filtre en quadrature')
```

Ensuite, pour pouvoir visualiser l'ondelette, le schéma suivant va permettre d'expliquer le processus :



Le premier étage contient deux points dits d'approximations du signal et chaque étage supérieur contient 2^J points du signal avec J qui est le numéro de l'étages. Ainsi, pour obtenir l'indice de la liste il suffit de respecter l'équation suivante :

$$i = 2^L + \sum_{J=1}^{Ligne-1} 2^J + Colonne$$

Avec « *Ligne* » l'indice de la ligne à laquelle se trouve l'ondelette à visualiser et « *Colonne* » l'indice de colonne dans laquelle se trouve l'ondelette à visualiser.

Il suffit alors de mettre en œuvre le code suivant :

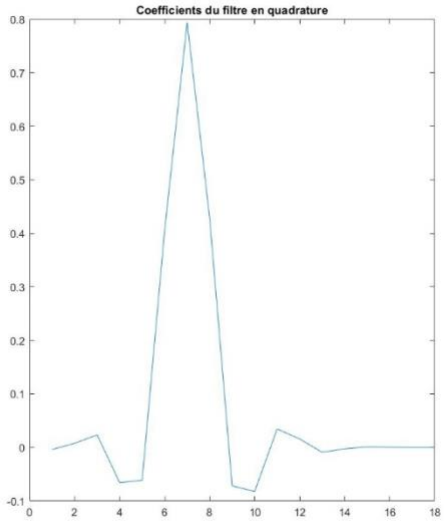
```
%2^J = taille du signal
J = 5;

%Creation de la liste des coefficients de la decomposition
en ondelette
wc1 = zeros(1,2^J);

%2^L donne le nombre de points d'approximations (J >> L)
L1 = 1;

%Paramètres libres : choix de l'ondelette
%L'étage de l'ondelette que l'on veut (a choisir)
Etage = 3;
%Colonne entre 0 et 2^Etage (a choisir)
Colonne = 7;

%Deduction de l'indice du coefficient concerne
indice = 2^L1;
for(i=1:Etage-1)
    indice = indice + 2^i;
end
indice = indice + Colonne;
```

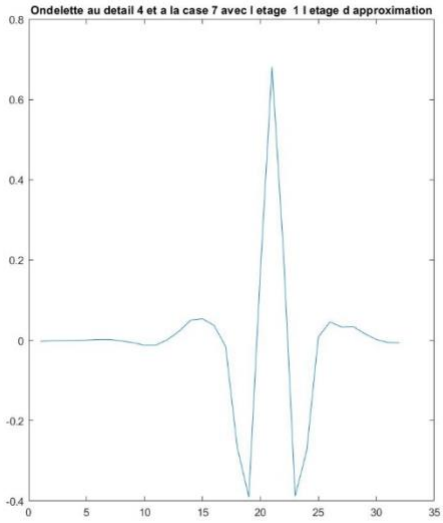
Une fois cet indice sélectionné, il faut alors considérer que chaque point central des cellules du tableau constitue un coefficient permettant de reconstruire un signal donné à l'aide d'une succession de filtrage (par les coefficients obtenus à partir des coefficients de quadrature obtenus précédemment). Aussi, si un des coefficients du tableau est mis à 0, alors l'ondelette n'est pas prise en compte lors de la reconstruction. Cette dernière remarque permet ainsi de reconstruire l'ondelette elle-même : en effet, en mettant tous les coefficients à 0 excepté le coefficient situé à l'indice de ligne et de colonne qui correspond à l'ondelette à visualiser qui lui est positionné à 1. Ceci permet alors de reconstruire l'ondelette. Pour cela, il faut ainsi exécuter le code suivant :

```
%Actualisation de la liste des coefficient (permet de
selectionner l'ondelette a
%obtenir apres reconstruction
wc1(indice) = 1;

%Reconstruction de l'ondelette
x1 = IWT_PO(wc1, L1, CoifQMF);

%Affichage de l'ondelette
subplot(122)
plot(x1)
```

```
title(['Ondelette au detail ', num2str(L1 + Etage), ' et a la case ', num2str(Colonne), ' avec l etage ', num2str(L1), ' l etage d approximation'])
```

Il est alors possible d'obtenir le résultat suivant :

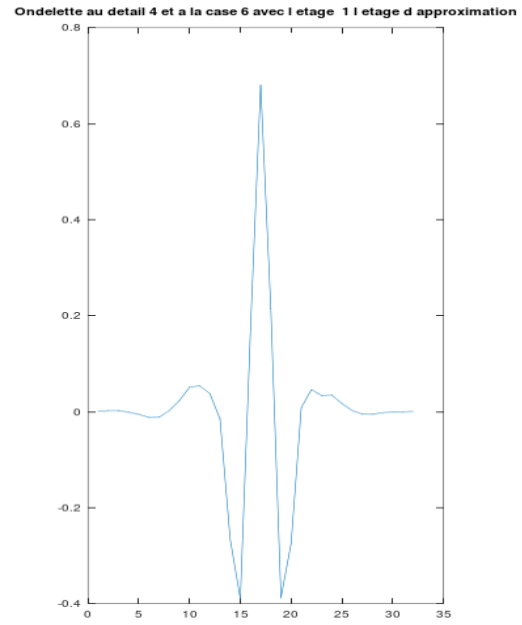


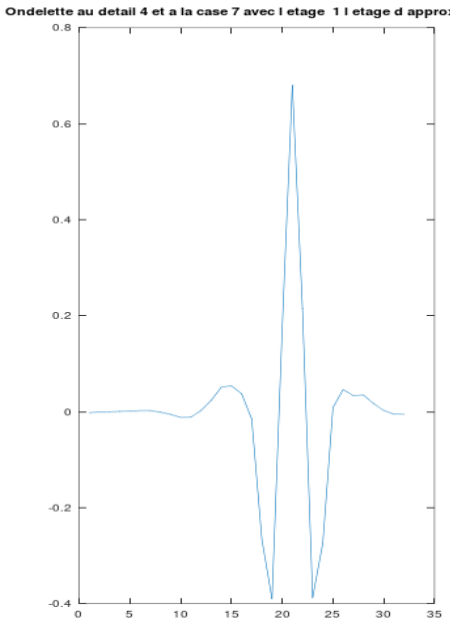


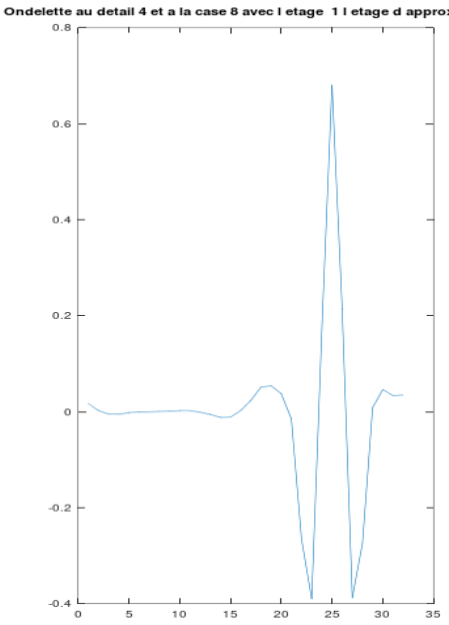
A droite, il est constatable que l'ondelette est de moyenne nulle puisque :

```
>> mean(x1)
ans =
  -1.7678e-13
```

Il est tout de même important de noter que le tableau précédemment évoqué influence beaucoup le comportement de l'ondelette. Aussi lorsque deux ondelettes successives sont visualisées, cela donne :







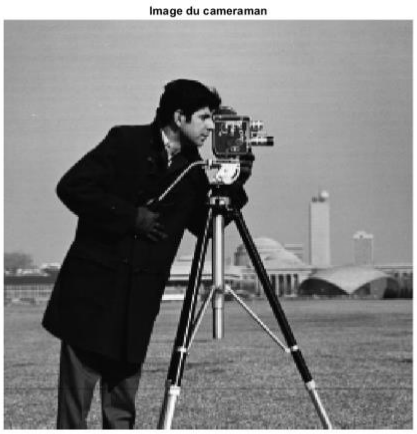
Il est alors remarquable que se déplacer d'une case dans le tableau ne permet pas de déplacer l'ondelette d'une unité mais plus exactement de $2^{\text{Ligne-1}}$. Ceci est dû au

principe même des ondelettes : plus la bande fréquentielle est élevée, plus la bande temporelle est étroite et inverse, ici puisque l'étude est à étage fixe, c'est la bande temporelle qui est modifiée et celle-ci relativement à sa hauteur dans le tableau.

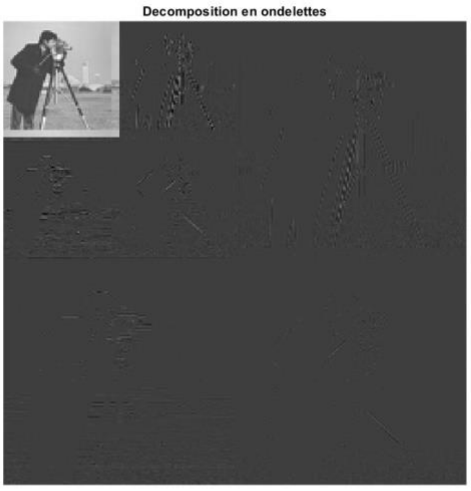
Maintenant que l'outil utilisé est plus clairement identifié mais aussi visualisé, il est possible d'appliquer la méthode de décomposition en ondelettes à des images. Pour cela, la fonction FWT2_PO est utilisée afin d'effectuer une analyse multi-résolution de cette image. Cette fonction est la même que celle d'avant hormis le fait que la première était destinée aux signaux à une dimension alors que la seconde permet le traitement des images (soit des signaux à deux dimensions). Mais le principe reste rigoureusement le même et le tableau reste valable également. Il est à noter que l'échelle la plus grossière entrée comme paramètre de la fonction va définir la taille de l'image la plus grossière. Cette image correspond au niveau d'approximation de la décomposition en ondelette, elle peut être directement reliée au paramètre L1 précédemment utilisé : il s'agit de l'image qu'il reste au moment où la décomposition s'est arrêtée. Pour réaliser l'analyse multi-résolution de l'image, il suffit alors de mettre en œuvre le code suivant :

```
%Affichage de l'image a traiter
figure(2)
I = imread('cameraman.tiff');
I = im2double(I);
imshow(I)
title('Image du cameraman')

%Decomposition en ondelette de l'image a traiter
L2 = 6;   %approximation : image de taille 2^L2 par 2^L2
pixels
wc2 = FWT2_PO(I, L2, CoifQMF);

%Affichage de la decomposition en ondelettes
figure(3)
subplot(121)
imshow(wc2, [])
title('Decomposition en ondelettes')

%Extraction de l'approximation de l'image
ImageGross = wc2(1:2^L2,1:2^L2);

%Affichage de l'approximation
subplot(122)
imshow(ImageGross, [])
title('Extraction image la plus grossiere')
```

La figure 2 permet de visualiser l'image du caméraman :



Et la figure 3 permet d'obtenir la décomposition en ondelettes de l'image et l'extraction de l'image la plus grossière qui est de taille $2\text{^}L2 * 2\text{^}L2$:



La décomposition permet d'observer la relation entre les fréquences et les ondelettes. En effet, dans le cas présent, la décomposition en ondelette s'est arrêtée très tôt soit, le L du tableau est assez proche du J maximum. Ainsi la décomposition est censée avoir traité uniquement les fréquences les plus hautes de l'image. Pour rappel, en imagerie, les hautes fréquences correspondent aux zones de l'image où les valeurs des pixels changent brusquement de valeurs, ce qui s'apparente très souvent aux contours d'un objet et aux délimitations entre deux couleurs, alors que les basses fréquences sont essentiellement représentées par des zones homogènes de l'image. Cette théorie est alors bien confirmée dans la décomposition en ondelette car les contours de l'homme et de sa caméra peuvent être devinés mais ceci est également visible sur l'approximation (l'image de droite). En effet, il apparait que les contours semblent être

floutés et beaucoup moins nettes qu'ils ne l'étaient sur l'image initiale, de la même manière, le visage de caméraman se confond en une seule masse et les détails de son visage ne sont plus visible sur cette approximation. Tout ceci caractérise particulièrement bien l'effet d'un filtrage de l'image : les différentes zones de la décomposition (bas droit, haut droit et bas gauche…) correspondent à une succession de filtrage passe haut et l'approximation correspond à un filtrage passe bas de toutes les harmoniques déjà présentes dans la décomposition.

Une fois cette décomposition effectuée, il est remarquable de pouvoir de nouveau reconstituer l'image et ce sans aucune perte d'information. En effet, à l'aide de la fonction IWT2_PO qui fait la transformation en ondelettes inverse, le résultat final est :

```
figure(4)
%Reconstruction de l'image
x2 = IWT2_PO(wc2, L2, CoifQMF);

%Comparaison entre l'image initiale et l'image reconstruite
subplot(121)
imshow(I)
title('Image originale')
subplot(122)
imshow(x2)
title('Image reconstruite')
```

À première vue, il semblerait que les images soient bien rigoureusement identiques, ce qui peut être confirmé à l'aide de la simple ligne de code suivante :

```
% %Verification de l egalite des figures
Norm = norm(x2-I);

>>> Norm =

   1.5677e-09
```

Ainsi les images sont bel et bien similaires puisque la norme de leur différence est nulle. Ainsi à l'aide des de l'analyse multi-résolution d'une image, Il est assez aisé de pouvoir reconstituer une image à partir d'une approximation, soit une image très légère en termes de mémoire, une liste de coefficients, propre à chaque ondelette utilisée, et les coefficients permettant le filtrage.

Ceci permet alors de déduire que la décomposition en ondelette peut être particulièrement efficace afin de compresser une image, en effet toutes les données n'ont pas à être stockées, il suffit de connaitre le type d'ondelettes utilisé, les coefficient du filtre ; toutes ces données étant fixes elle peuvent être pré-implémentée et n'ont pas besoin d'être transmises, et ainsi la transmission ne nécessite que la liste des coefficients de détails ainsi que l'approximation de l'image.

1.2 Application au débruitage d'une image

Dans cette partie, le but premier est de trouver une nouvelle application à la décomposition en ondelettes, à savoir, le débruitage d'image.

Pour tester cette application, l'utilisation d'une image non bruitée va permettre d'être l'image témoin et donc de comparer si une fois débruitée l'image finale se rapproche de la vraie image. Ainsi, la première étape consiste évidemment à bruiter l'image initiale :

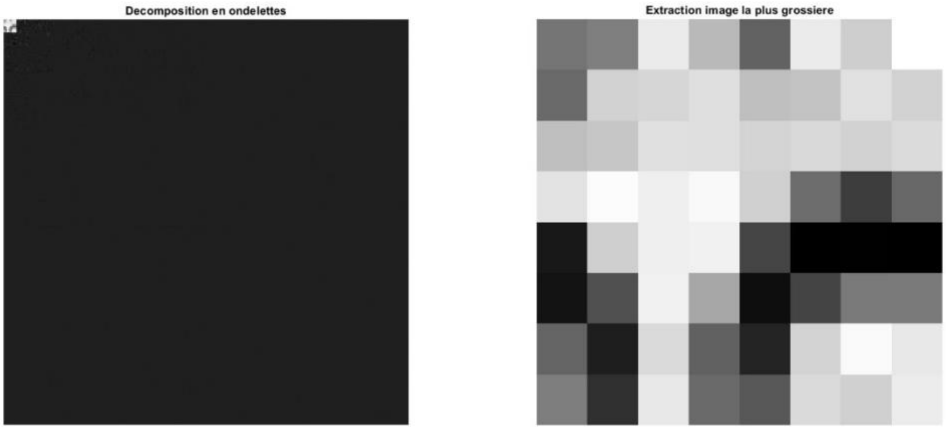
```
figure(5)
Ingrid = ReadImage('Daubechies');
subplot(221)
imshow(Ingrid,[])
title('Image originale')

NoisyIngrid = Ingrid + 5*WhiteNoise(Ingrid);
```

Puis il suffit d'amorcer la décomposition multi-résolution de cette image bruitée à l'aide de l'ondelette Coiflet jusqu'à l'échelle 3 (soit obtenir une approximation de 8px/8px) et ce de la même manière que précédemment.

```
L3 = 3;
wc3 = FWT2_PO(NoisyIngrid, L3, CoifQMF);
```

```
figure(6)
subplot(121)
imshow(wc3, [])
title('Decomposition en ondelettes')

ImageGross2 = wc3(1:2^L3,1:2^L3);

subplot(122)
imshow(ImageGross2, [])
title('Extraction image la plus grossiere')
```

Ce ui permet l'obtention de la décomposition et de l'approximation suivante :



On observe alors que l'image la plus grossière est complètement destructurée et qu'il n'y a plus aucune possibilité de distinguer le portrait initialement présent sur l'image. Et pourtant, cette petite portion d'image va permettre une reconstruction totale de l'image.

Il est à noter que pour la suite de cette étude, l'approximation a été préservée dans une variable à indépendante du résultat de la décomposition en ondelettes. En effet, l'étape suivante consiste en un seuillage de la décomposition en ondelettes. Le bruit étant une variation brusque d'un pixel à l'autre, il est donc associé à des hautes fréquences, en seuillant ces hautes fréquences, ceci va permettre d'éliminer les changements de variation beaucoup trop brusques ou au pire de les atténuer fortement. Pour ne pas perdre trop d'informations, notamment concernant les contours, le seuillage doux est ici privilégié (à l'aide de la commande `SoftThresh`).

Il est à noter que l'approximation elle ne doit pas subir un tel seuillage, en effet, celle-ci correspond à toutes les fréquences n'ayant pas encore été traitée, soit, il s'agit essentiellement des basses fréquences de l'image, or comme évoqué précédemment, le bruit ne se situe pas dans les basses fréquences mais dans les hautes fréquences. Seuiller l'approximation engendrerait une grosse perte d'information sans aucun bénéfice sur le résultat final.

```
seuillage = SoftThresh(wc3,10);
seuillage(1:2^L3,1:2^L3) = ImageGross2;
```

Puis il suffit d'appliquer la méthode vue dans la partie précédente qui consiste à utiliser la fonction IWT2_PO faisant la transformation en ondelettes inverse pour reconstruire l'image et qui prend en argument la matrice de décomposition en ondelettes `wc3`, l'échelle la plus grossière `L3` et le filtre de quadrature `CoifQMF`.

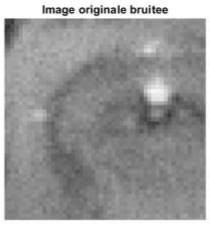
Désormais il ne reste plus qu'à afficher les images afin de pouvoir réellement les comparer et statuer sur l'efficacité de cette méthode.

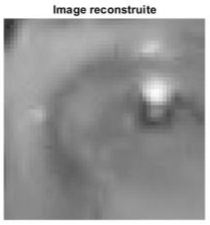
```
figure(5)
subplot(223)
imshow(NoisyIngrid,[])
title('Image originale bruitee')

x3 = IWT2_PO(seuillage, L3, CoifQMF);
subplot(224)
imshow(x3,[])
title('Image reconstruite')
```

Il est donc remarquable que l'image bruitée a été correctement reconstruite même si cette opération flouter certains détails présents dans les cheveux ou encore au niveau des lunettes, cette perte pouvait être prédite de par le fait que la « texture » des cheveux n'est pas du tout homogène : elle relève des hautes fréquences, et le seuillage a donc nécessairement engendré une perte d'information à ce niveau là.

Ces dernières remarques sont d'autant plus flagrantes à petite échelle. Afin d'illustrer ceci, il est intéressant de se placer sur la région autour de l'œil à l'aide des quelques lignes d'ajout suivantes :

```
axis([110 160 110 160]);
axis square
```





L'observation tend alors à confirmer les premières impressions : l'image a bien été débruitée et reconstruite (plus de pixels parasites) mais l'image reconstruite reste tout de même floue pour les mêmes raisons évoquées sur la figure précédente.

Ainsi, l'analyse multi-résolution suivie d'un seuillage est une bonne méthode pour débruiter et reconstruire une image mais la perte d'information reste quand même présente et se traduit par une image plus floue que d'ordinaire.

2. Classification : transformée en ondelettes continues

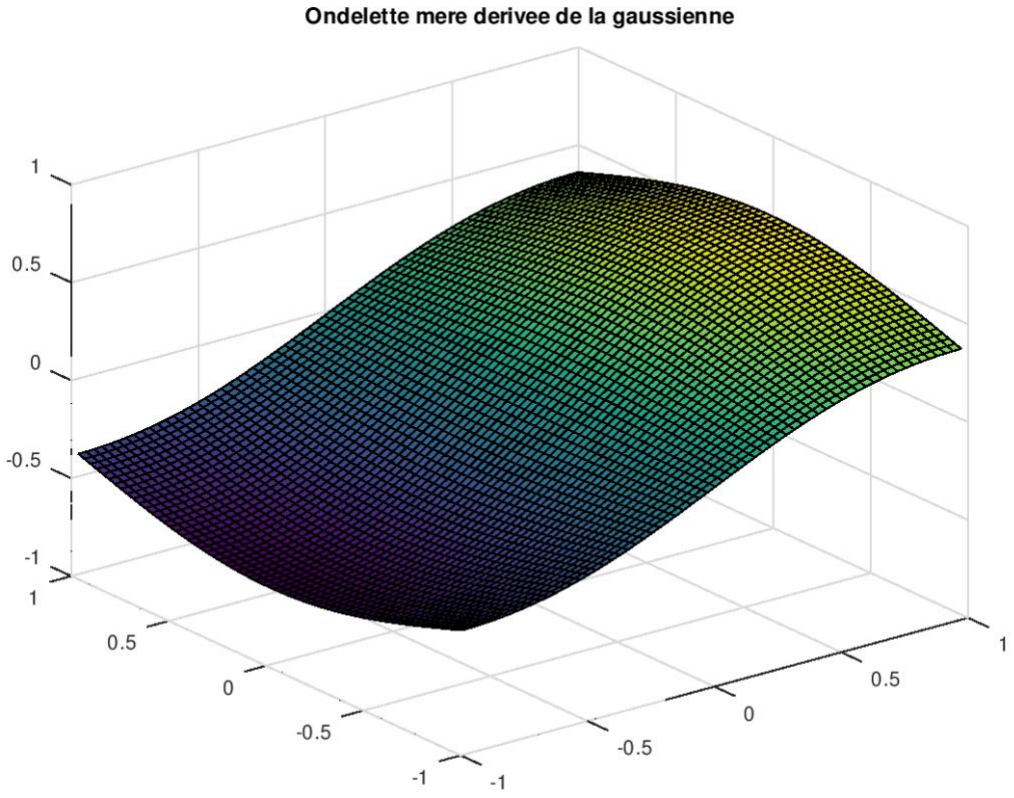
L'application des ondelettes visées dans cette dernière partie consiste en la caractérisation de textures et donc la classification de celle-ci. Pour se faire, l'utilisation d'ondelettes discrètes ne suffit plus et se sont des ondelettes continues qui vont alors servir ici.

2.1 Ondelettes orientées : dérivées d'une gaussienne

Afin de mettre en œuvre cette classification, la création des ondelettes gaussiennes est alors requise. Pour se faire, tout commence par une ondelette mère : la dérivée d'une gaussienne, la mise en place d'une telle ondelette se fait comme suit :

```
Nx = 64;
Ny = 64;
x = -1+1/Nx:2/Nx:1-1/Nx;
y = -1+1/Ny:2/Ny:1-1/Ny;
[X,Y] = meshgrid(x,y);

%Creation de l'ondelette derivee de la gaussienne
```

```
phi = X.*exp(-(X.^2+Y.^2)/2);
surf(X,Y,phi)
title('Ondelette mere derivee de la gaussienne')
```



Il est ici facile de se convaincre que cette courbe correspond bien à la dérivée de la gaussienne de par son maxima et son minima qui sont caractéristiques de cette fonction dérivée.

Malgré tout, une unique ondelette ne suffit pas à caractériser une image complètement, en effet, le résultat de la transformée en ondelette d'une image est bien différent selon l'orientation de cette ondelette mais aussi selon l'échelle utilisée.

Aussi afin de créer des ondelettes filles, il est nécessaire de réaliser une rotation de l'ondelette mère mais également un changement d'échelle à l'aide de la formule suivante :

$$\begin{cases} \Theta(X,Y) = e^{-\frac{(\frac{X}{2^j}+\frac{Y}{2^j})}{2}} \\ \Psi(X,Y) = \frac{1}{2^j} * \left(\frac{X}{2^j}\cos(\alpha) + \frac{Y}{2^j}\sin(\alpha)\right) * \Theta(X,Y) \end{cases}$$

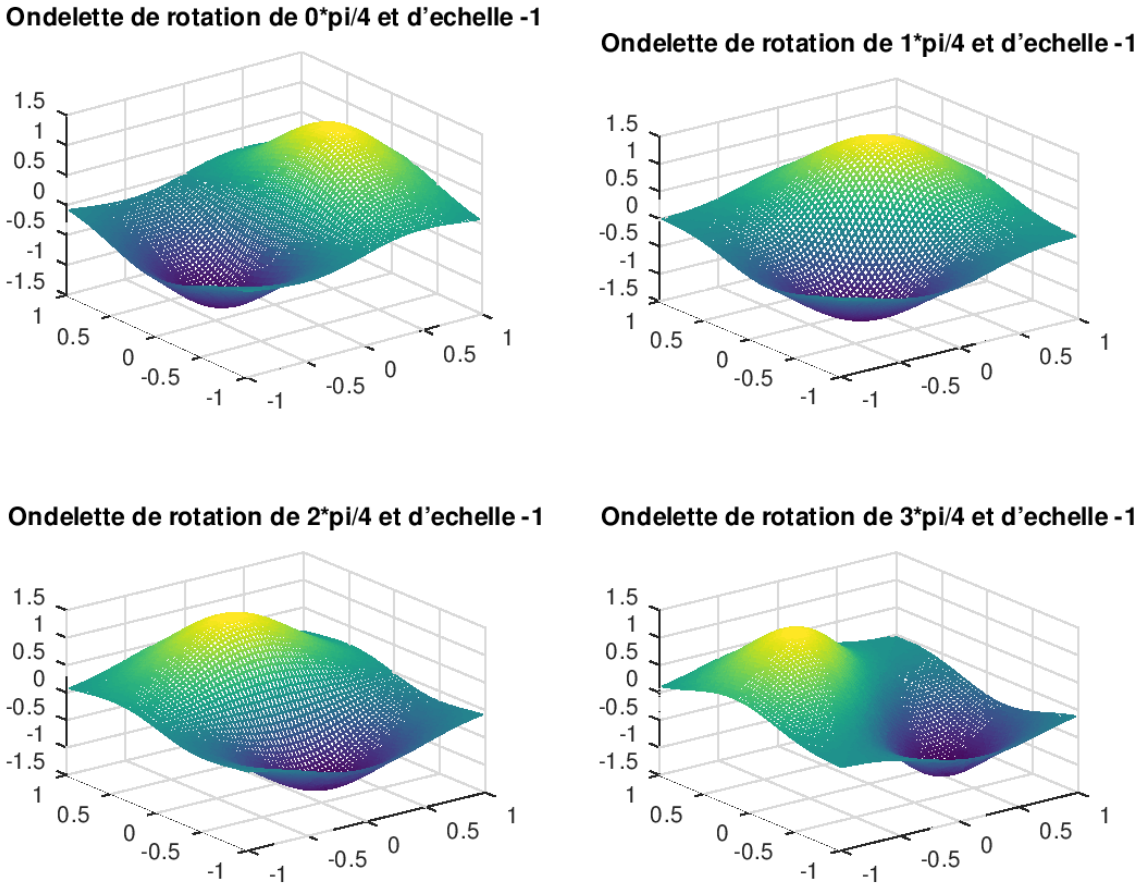
Avec :

- j la puissance permettant le changement d'échelle
- α l'angle de rotation de l'ondelette
- Θ la gaussienne ayant subi un changement d'échelle
- Ψ la dérivée de la gaussienne ayant subi un changement d'échelle et une rotation (soit l'ondelette à visualiser)

Le code permettant de réaliser une telle opération est alors le suivant :

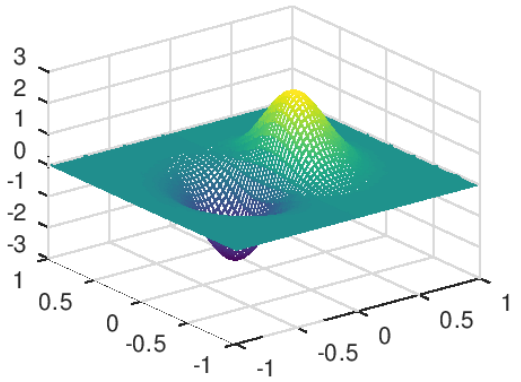
```
%Parcours des differentes echelles
for j=-4:1:-1
    i = 1;
    %Parcours des differents angle de rotation
   for theta = 0:pi/4:pi-pi/4
        %Changement d'echelle
        Xj = X/(2^j);
        Yj = Y/(2^j);
        %Calcul de la gaussienne
        T = exp((-1/2) * (Xj.^2 + Yj.^2));
        %Calcul de sa derivee combinee au
changement d'echelle et a une rotation
        psij = (1/2^j) * (Xj*cos(theta) +
Yj*sin(theta)).* T;

        %Affichage des ondelettes filles
        figure(7 - j)
        subplot(2,2,i)
        mesh(X,Y,psij)
        title (['Ondelette de rotation de ',
num2str(i-1),'*pi/4 et d''echelle ', num2str(j)]);
    end
end
```

Ce qui permet d'obtenir les nombreux résultats suivants :

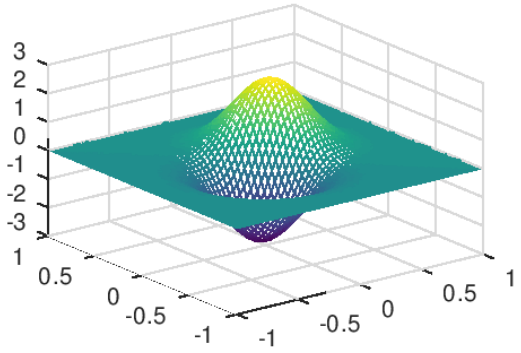


Cette première image permet de visualiser la façon dont l'ondelette évolue en fonction d'une rotation, il parait alors évident que chaque ondelette ne permettra pas de détecter les mêmes informations selon l'angle de rotation qu'elle a connu. Cette notion peut être mise en parallèle avec le calcul du gradient d'une image. En effet, le masque utilisé pour calculer le gradient horizontal et vertical d'une image (tel que le masque de Sobel) est très similaire, seule l'orientation de celui-ci diffère. Et ce simple changement d'orientation permet dans le premier cas de détecter l'ensemble des contours verticaux de l'image et dans le second cas l'ensemble des contours horizontaux de l'image. L'idée est ici rigoureusement la même, selon l'orientation dans laquelle se trouve l'ondelette, elle va alors détecter plus ou moyen bien les fréquences à caractériser, c'est pourquoi utiliser plusieurs orientations permet d'extraire un maximum d'information de l'image.

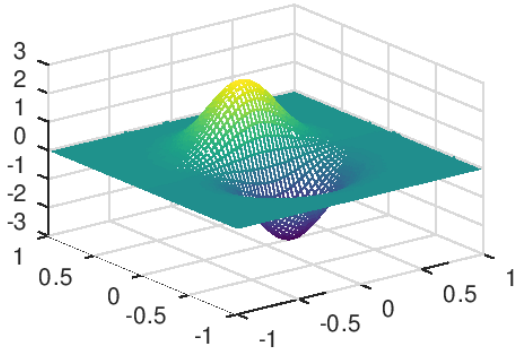
Ondelette de rotation de 0*pi/4 et d'echelle -2



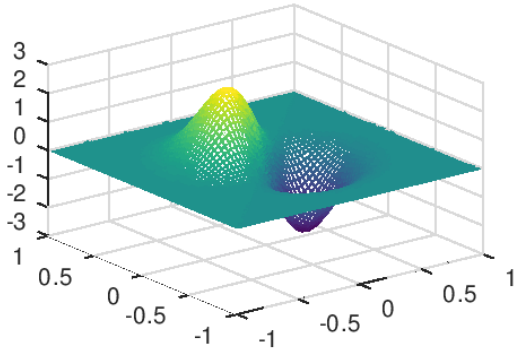
Ondelette de rotation de 1*pi/4 et d'echelle -2



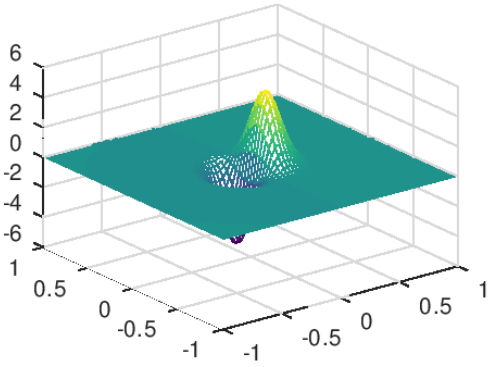
Ondelette de rotation de 2*pi/4 et d'echelle -2



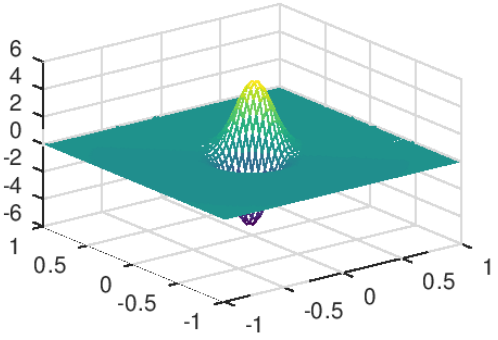
Ondelette de rotation de 3*pi/4 et d'echelle -2



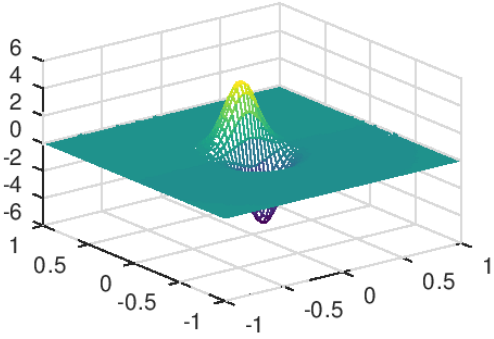
Ondelette de rotation de 0*pi/4 et d'echelle -3



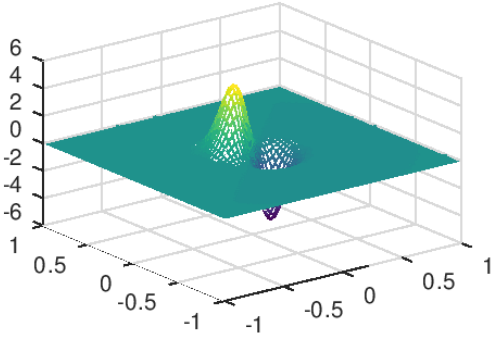
Ondelette de rotation de 1*pi/4 et d'echelle -3

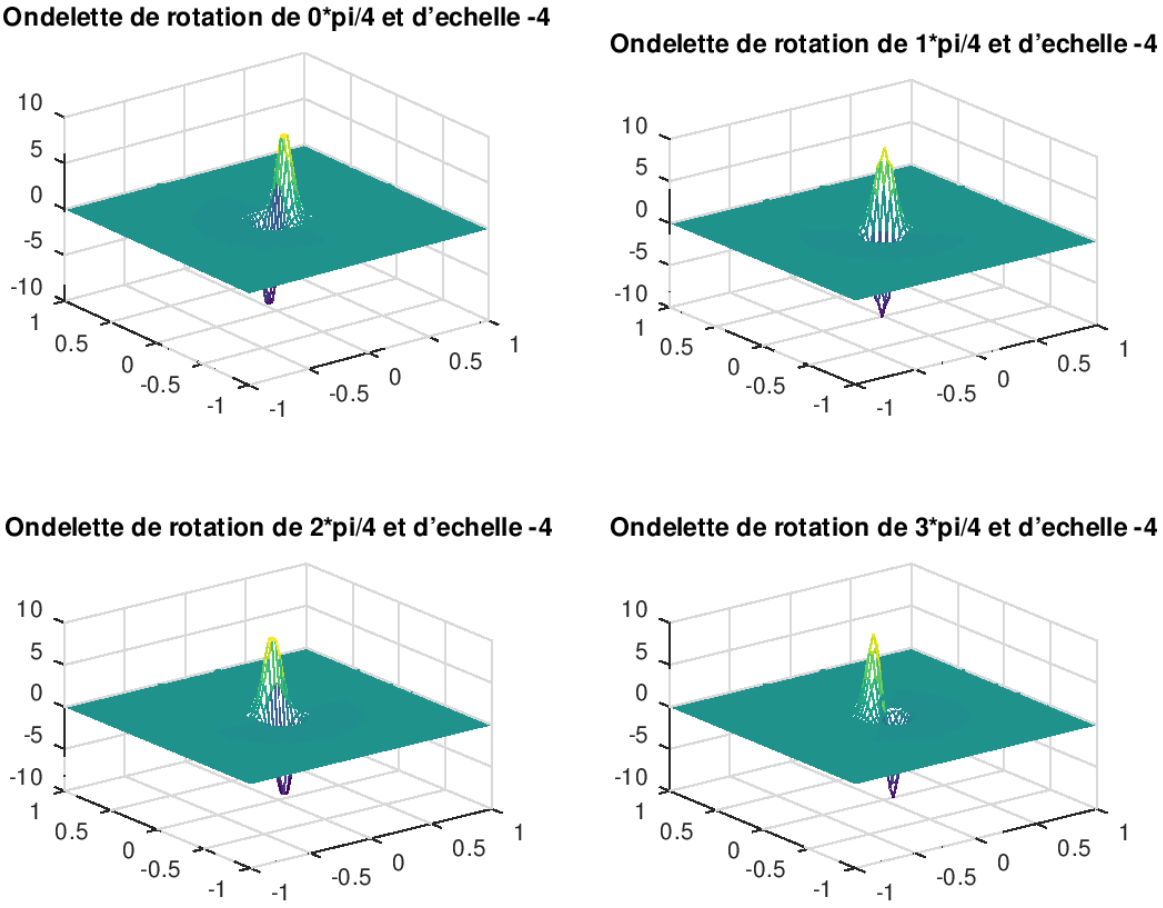


Ondelette de rotation de 2*pi/4 et d'echelle -3



Ondelette de rotation de 3*pi/4 et d'echelle -3



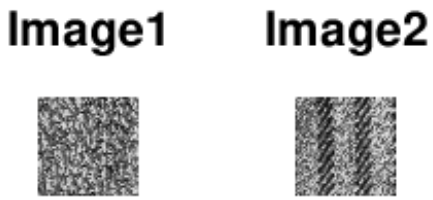


Ces trois dernières figures permettent de visualiser l'évolution de l'ondelette dès qu'un changement d'échelle lui est appliqué. Pour rappe c'est ce changement d'échelle qui est à la base même de la théorie des ondelettes et qui la rend particulièrement consistante : celui-ci permet de balayer toutes les fréquences (cf tableau dans la toute première partie). Ainsi donc, les changement d'échelles vont permettre de parcourir les fréquences présentes dans l'image par ordre croissant de fréquence (la dernière figure expose une fréquence plus importante que troisième, …, jusqu'à la première qui témoigne d'une fréquence assez basse).

Ainsi le changement d'orientation associé à l'analyse multi-résolution des images va permettre de particulièrement bien les caractériser et ainsi de pouvoir avoir plusieurs facteurs de comparaison afin de classifier les textures.

2.2 Application à la caractérisation de textures

Cette application vise à ranger deux images dans des catégories de textures pré-établies suivantes :



Ainsi, le bon sens permet de déduire assez rapidement que l'image 1 rentre dans la catégorie de la texture 7 et que l'image 2 rentre dans la catégorie de la texture 5. Il reste désormais à montrer algorithmiquement parlant que cette solution convient.

Pour se faire, il faut décomposer le travail en plusieurs étapes :

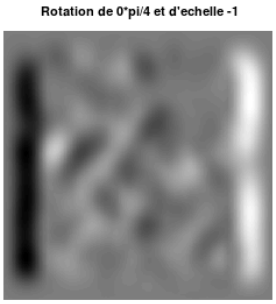
- Faire la transformée en ondelette de chaque texture et de chaque image
- Calculer le vecteur caractéristique de chacune (soit calculer la norme au carré de la transformée en ondelette)
- Prendre la valeur absolue de la différence entre chaque vecteur caractéristique de texture et de l'image 1 et chaque vecteur caractéristique de texture et l'image 2
- Rechercher les textures minimisant chaque la différence précédente pour chaque image
- Rechercher quelle texture minimise le plus de différence

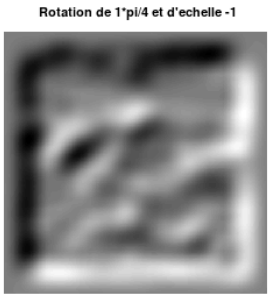
Ceci se traduit algorithmiquement parlant par les lignes de code suivantes :

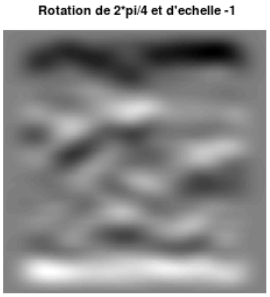
```
vects = [] ;
for nbfig = 1:9
    %Importation du fichier
    if nbfig <= 7
        text1 =
im2double(imread(['Texture',num2str(nbfig),'.png']));
    else
        nb = nbfig-7;
        text1 =
im2double(imread(['Image',num2str(nb),'.png']));
```

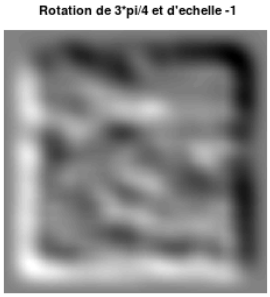
```
    end

    vectcaract = [];

    %Parcours des differentes echelles
    for j=-4:1:-1
        i = 1;
        %Parcours des differents angle de rotation
        for theta = 0:pi/4:pi-pi/4
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
            %% Creation d'une nouvelle ondelette psij %%
            %%      (Voir code precedent)             %%
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

            %Transformee en ondelette de la texture (par la
convolution)
            prod = conv2(text1,psij);

            %Actualisation du vecteur caracteristique
            vectcaract = [ vectcaract,  norm(prod)^2 ];

            i = i+1;
        end
    end

    %Actualisation de la matrice contenant l'ensemble des
vecteurs caracteristiques par ligne
    vects = [vects ; vectcaract];

end

%Comparaison entre les textures et le images
vect1 = abs(vects(1:7,:) - ones(7,1) *vects(8,:));
vect2 = abs(vects(1:7,:) - ones(7,1)* vects(9,:));

%Isolation des valeurs minimales
[Im1, pos1] = min(vect1);
[Im2, pos2] = min(vect2);

%Recherche des texture correspondantes aux images
Sim1 = mode(pos1);
Sim2 = mode(pos2);
```

Ainsi lors de l'affichage des résultats des différentes transformées en ondelettes sur la première texture, ceci permet d'obtenir les résultats suivants :

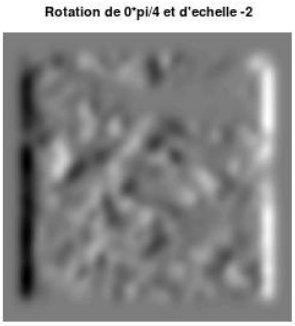


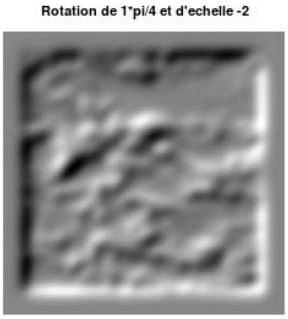


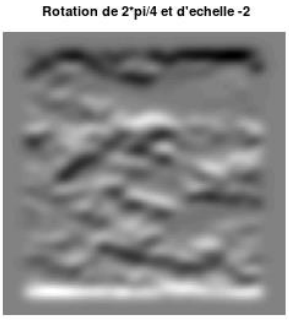


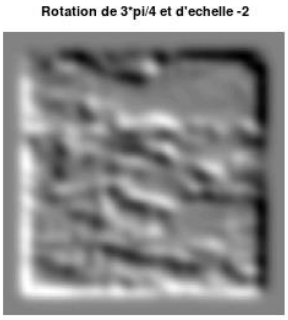


Cette figure reflète parfaitement la remarque effectuée sur l'effet de la rotation de l'ondelette sur l'obtention d'images parfaitement distinctes les unes des autres. En effet chacune des quatre images étant issue d'une ondelette dont la rotation diffère, les zones de l'image ciblées et retranscrites lors de la transformée en ondelette sont ainsi différentes et pourtant complémentaires car elles sont issues de la même image. Aussi, chaque image possède une information supplémentaire que les autres.

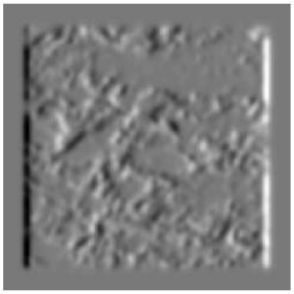




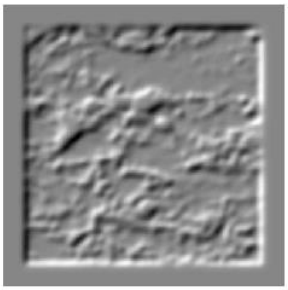




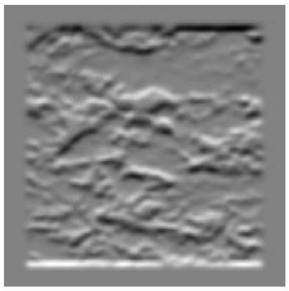
Rotation de 0*pi/4 et d'echelle -3



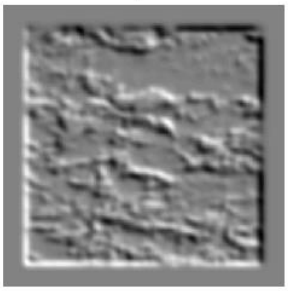
Rotation de 1*pi/4 et d'echelle -3



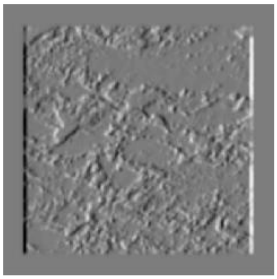
Rotation de 2*pi/4 et d'echelle -3



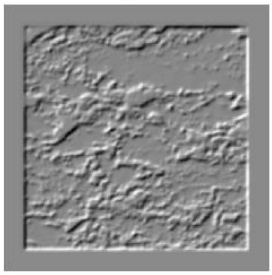
Rotation de 3*pi/4 et d'echelle -3



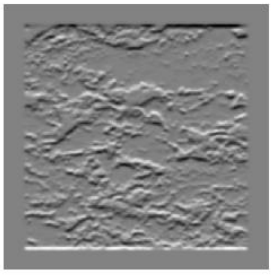
Rotation de 0*pi/4 et d'echelle -4



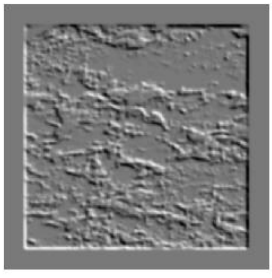
Rotation de 1*pi/4 et d'echelle -4



Rotation de 2*pi/4 et d'echelle -4



Rotation de 3*pi/4 et d'echelle -4



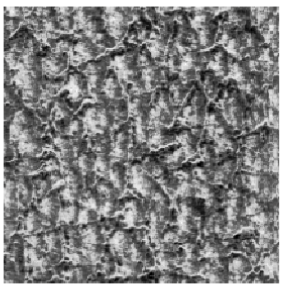
L'observation de ces dernières images permet de confirmer la seconde remarque effectuée plus haut concernant les ondelettes fille : plus le changement d'échelle est important et plus les détails de l'image sont visibles. Ceci est caractéristique à une transcription de hautes fréquences, soit, plus le changement d'échelle est élevé, plus l'ondelette permet de détecter des hautes fréquences. Plus le changement d'échelle est faible et plus il se rapproche des basses fréquences.

En parallèle de toutes ces constatations, cette étude visait à classer l'image 1 et l'image 2 dans une catégorie de textures, aussi suite à l'algorithme précédent, les positions des minima relatifs à l'image 1 et à l'image 2 sont repérés grâce aux variables pos1 et respectivement pos2 :

```
>> pos1
pos1 =

   5    7    5    5    7    7    5    7    7    7    5    7    7    7    5
7

>> pos2
pos2 =

   5    5    5    5    5    5    5    5    5    5    5    3    4    4    5
4
```

Il semblerait alors que l'algorithme permette alors de bien classifier les images dans les textures qui leur correspond, car en effet, lorsque la texture visée par Sim1 et Sim2 sont affichées, ceci permet d'obtenir :

Image 1



Texture correspondante a l'image 1

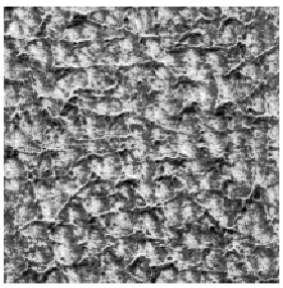
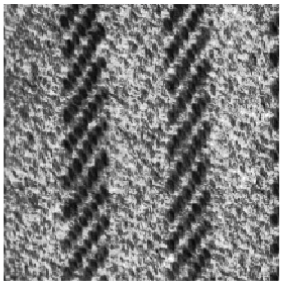
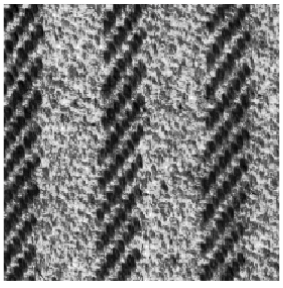


Image 2



Texture correspondante a l'image 2



Ainsi cette étude permet de démontrer que la transformée en ondelette peut également permettre de caractériser des images et ainsi donc de pouvoir les classifier dans différentes catégories distinctes.

Conclusion générale :

Cette étude a permis d'illustrer les principales applications de la décomposition en ondelette et de la transformée en ondelettes. Ainsi 4 grandes facettes de l'imagerie peuvent être traiter à l'aide d'ondelettes : la compression d'image, la reconstruction d'image, le débruitage d'image et la classification d'image. Tout ceci permet alors de statuer quant à la puissance de l'outil que sont les ondelettes ce qui explique pourquoi elles sont autant utilisées dans cette discipline.