

Comparaison des méthodes de débruitage

Rapport de projet

Réalisé par :

BEN HAMOUDA Amel
DURAND Aurélien

Table des matières

Résumé :.....	3
I. Introduction.....	3
1. Présentation du problème.....	3
2. Méthodes proposées.....	3
II. Wavelet.....	3
1. Analyse théorique.....	3
2. Constatations numériques.....	6
3. Estimation de la variance du bruit.....	10
4. Critiques.....	10
III. NL Means.....	11
1. Analyse théorique.....	11
A - Niveau pixel.....	11
B - Niveau patch.....	12
C – Avantage et inconvénient des deux niveaux.....	12
2. Constatations numériques.....	12
3. Estimation de la variance du bruit.....	15
4. Critiques.....	15
IV. Conclusion.....	16
V. Bibliographie.....	16
VI. Glossaire.....	17
VII. Annexe.....	17

Résumé :

Le problème étudié dans ce papier est la comparaison des méthodes de débruitage. Ce problème est plus ou moins pertinent puisqu'il permet de comprendre et de connaître les méthodes les plus importantes de débruitage pour optimiser le code dans un projet qui nécessite du débruitage. Ce qui permettra d'avoir un code à faible complexité et bien optimisé.

Pour répondre à ce problème on va dans la suite de ce document étudier deux des plus grandes méthodes de débruitage, le **débruitage par ondelette (Wavelet)** et le **débruitage en NLMeans**.

Pour étudier ce problème on a développé sur notebook jupyter (python) les différentes méthodes.

Mot clés : Wavelet, NLMeans, Hard Thresholding, Soft Thresholding, SNR.

I. Introduction

1. Présentation du problème

Le débruitage est un problème assez classique en traitement d'image et survient de différentes façon en fonction du capteur utilisé. Il existe ainsi différents types de bruits et chacun doit être traité spécifiquement. Dans ce papier nous avons voulu analyser et comprendre deux types de débruitage à savoir un débruitage par Ondelettes et un Débruitage utilisant une méthode non-local appelée NL-means. Le bruit utilisé est alors un bruit gaussien afin de pouvoir comparer les deux méthodes. Le débruitage consiste alors à atténuer le bruit présent dans un signal à la sortie d'un capteur en appliquant une succession d'opération ayant pour but de s'approcher aux mieux d'une image en l'absence de bruit. Cependant ce problème est non trivial et amène un certain nombre de complications tel que la disparition des bords d'une image, la perte d'information en fonction du filtrage réalisé...

2. Méthodes proposées

Pour répondre aux problèmes du débruitage nous nous sommes donc concentré sur deux débruitage différents.

Le premier est un débruitage utilisant les transformées en ondelettes basé principalement sur le travail trouvé sur le lien [suivant](#).

La seconde méthode est un débruitage techniquement non-local travaillant avec des patches de pixels qui se ressemblent afin de moyenniser l'information. Cette technique utilise une méthode appelée NL-means et nous nous sommes principalement basé sur l'article de *Antoni Buades et al.*

II. Wavelet

1. Analyse théorique

Les Ondelettes "Wavelets" sont un codage en sous bande d'une image consistant à extraire les caractéristiques d'une image sur plusieurs niveau d'échelle. Le principe est de décomposer l'image en haute et base fréquence en sous samplant les résultats de filtre passe-haut et passe-bas. Cette transformation est unitaire et repose sur la décomposition de base orthonormales et si l'on note O cette transformation on a : $O^t O = \text{Identité}$, $O^{-1} = O$

Ainsi pour un signal x que l'on visualise et en se plaçant dans une optique de dé-bruitage on aura :

$$y = x + b$$

Et :

$$Oy = Ox + Ob$$

On peut alors travailler dans le domaine des ondelettes pour extraire le bruit. La transformée en ondelette représente alors la transformation suivante sur f :

$$W_{\psi} f(j, k) = \langle f, \psi_{j,k} \rangle$$

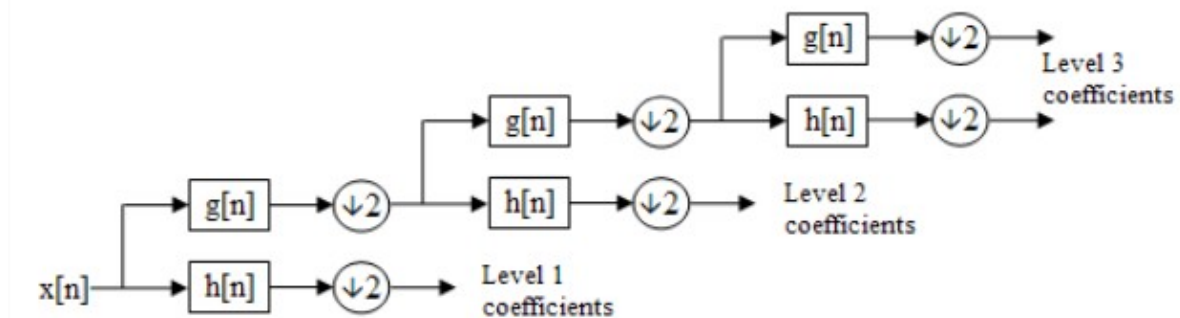
Où les $\psi_{j,k}$ sont une familles de base orthonormée de $L^2(\mathbb{R})$, j correspond aux niveau d'échelle souhaité.

$$\psi_{j,k}(t) = 2^{-j/2} \psi(2^{-j}t - k)$$

$f(t)$ est donc de la forme :

$$f(t) = \sum_{j \in \mathbb{Z}} \sum_{k \in \mathbb{Z}} \langle f, \psi_{j,k} \rangle \psi_{j,k}$$

Les signaux étant des images on se place dans un cadre discret. L'application successive des opérations de filtrage passe-bas et passe-haut donne en 1D le schéma suivant :



Les filtres $h[n]$ dénote un filtrage passe-haut tandis que les filtres $g[n]$ sont des filtres passe-bas. Pour chaque filtrage on applique ensuite l'opération de sous échantillage ($\downarrow 2$) consistant donc à récupérer un coefficient sur deux du résultats de filtrage afin de créer un nouveaux sous espace et ré-appliquer une ondelette.

Au passage en 2D le schéma devient alors :

C_{j-2}	D_{j-2}^1	D_{j-1}^1
D_{j-2}^2	D_{j-2}^3	
D_{j-1}^2		D_{j-1}^3

Ici on doit donc travailler sur les colonnes et lignes de l'images et ainsi on donne alors pour 1 niveau d'échelle :

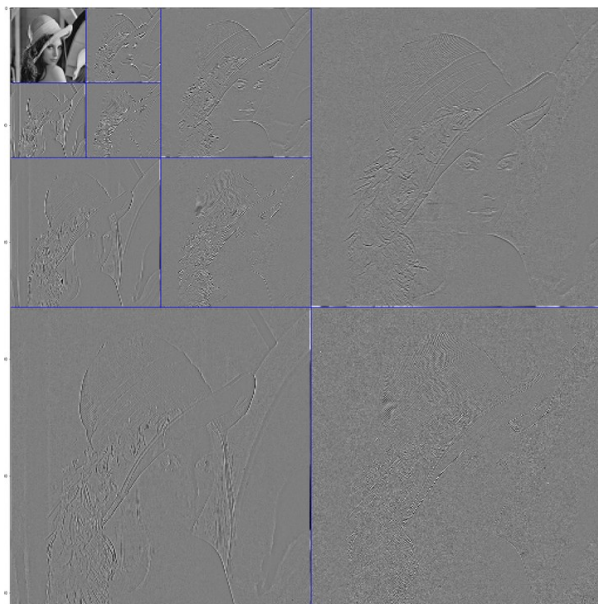
D_{j-1}^3 : Application de passe haut en ligne et en colonne

D_{j-1}^2 : Application de passe haut en ligne et passe-bas en colonne

D_{j-1}^1 : Application de passe-bas en ligne et passe haut en colonne

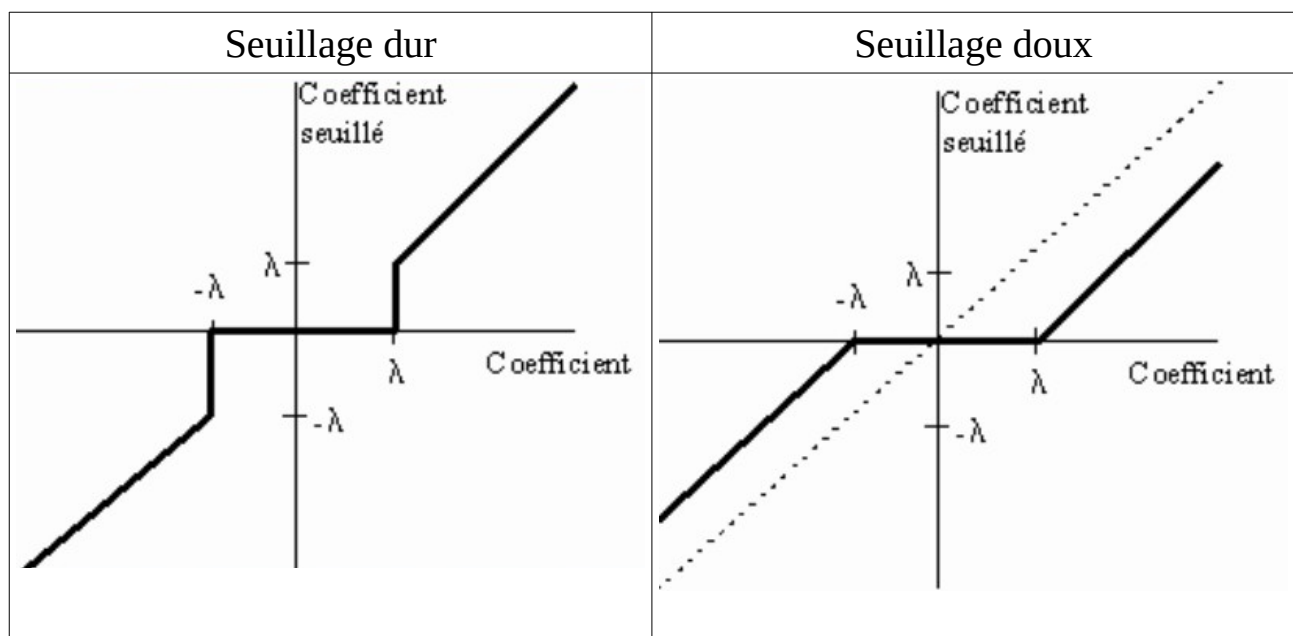
C_{j-1} : Application de passe bas en ligne et en colonne

Une représentation sur une image, d'une transformé en ondelettes, donne alors :



Ici on est sur 2 niveaux de sous échantillonnage ($j = 2$). L'information principale de l'image est contenue dans la partie basse fréquence en haut à gauche et plus l'on remonte les niveaux plus on a de l'information haute fréquence.

Afin d'utiliser les ondelettes pour faire du débruitage ("denoising") on vient appliquer un seuil ("Threshold") sur l'image d'ondelette. On procède ainsi car dans une image si l'on suppose un bruit blanc, celui-ci va se répartir sur toutes les fréquences mais sera bien plus impactant en haute fréquence. De ce fait, en venant appliquer un seuil on peut éliminer une partie du bruit. En effet dans l'image les coefficients basses fréquences qui comporte donc l'information principale de l'image sont les coefficients avec un plus grand niveau de gris. Tandis que l'information haute fréquence est caractérisée par les plus petits niveaux de gris. Appliquer un seuil va avoir pour conséquence d'éliminer une partie du bruit mais va aussi affecter les parties hautes fréquences qui représentent les bords/contours de l'image. Pour le Threshold il y a alors différentes manières de procéder, par exemple par soft ou par hard Threshold (seuillage doux/dur).



Une fonction de seuillage dur sera de la forme suivante :

$$\begin{cases} x_T(t) = x(t) & \text{si } x(t) > T \\ x_T(t) = 0 & \text{sinon} \end{cases}$$

Et une fonction de seuillage doux :

$$\begin{cases} x_T(t) = x(t) - \text{sign}(x(t)) * T & \text{si } x(t) > T \\ x_T(t) = 0 & \text{sinon} \end{cases}$$

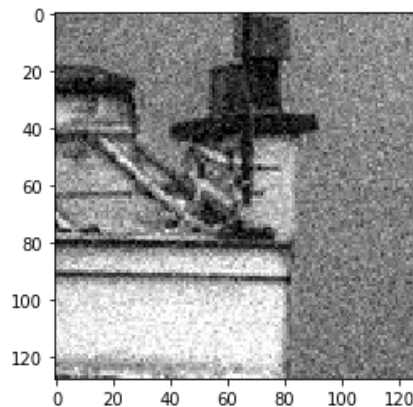
Le seuillage doux permettant d'obtenir une image plus régulière. Ici on utilise donc le fait que la transformée en ondelette préserve l'énergie du signal. Ainsi dans le signal les grand coefficient avec la plus grande énergie représente le signal tandis que le bruit, malgré qu'il impact tout le signal, va majoritairement être représenté par les plus les petits coefficients de la transformée à plus faible énergie. D'où le fait qu'il faille seuiller la transformée dans une optique d'élimination du bruit.

2. Constatations numériques

Sur python on a regardé l'effet du débruitage dans le domaine des ondelettes par seuillage dur et par seuillage doux. L'image de référence est la suivante :



On commence par y ajouter du bruit :



On a pris un bruit blanc gaussien de variance égal à 20.

Pour les ondelettes les conditions sur les filtres sont les suivantes :

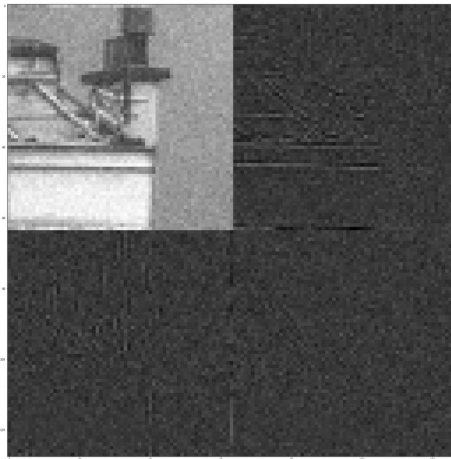
Sur le passe-bas :

$$\begin{cases} |\hat{h}(\omega)|^2 + |\hat{h}(\omega + \frac{1}{2})|^2 = 2 \\ \hat{h}(0) = \sqrt{2}. \end{cases}$$

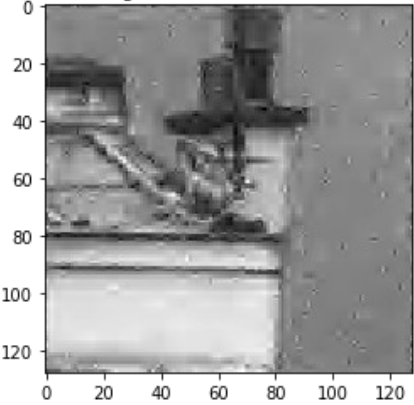
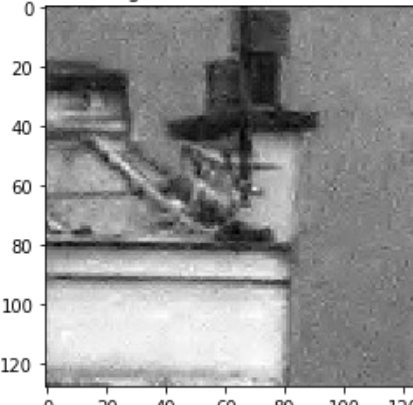
Sur le passe-haut :

$$\begin{cases} |\hat{g}(\omega)|^2 + |\hat{g}(\omega + \frac{1}{2})|^2 = 2 \\ \hat{g}(\omega)\hat{h}^*(\omega) + \hat{g}(\omega + \frac{1}{2})\hat{h}^*(\omega + \frac{1}{2}) = 0. \end{cases}$$

On décompose alors l'image sous forme d'ondelette avec l'ondelette de Daubechies et on obtient pour 1 niveau :



Ensuite on applique les opérateurs de base de soft et hard Threshold défini auparavant et en reconstruisant l'image on obtient :

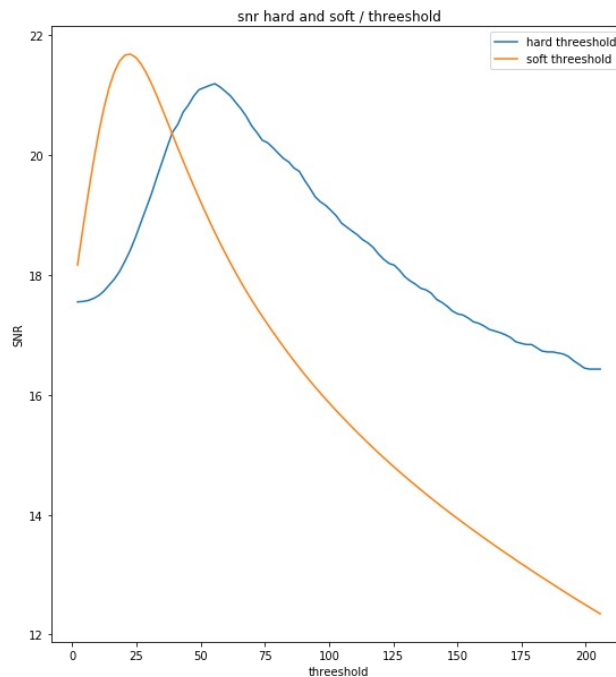
Seuillage dur	Seuillage doux
<p>Hard threshold image with threshold:53.465207187044825</p> 	<p>Soft threshold image with threshold:20.56354122578647</p> 

Dans le cas du hard Threshold on observe quasiment plus de bruit mais cependant il y a un grand nombre d'artefact du fait du passage "brutal" de 0 aux valeurs de seuil, un seuillage hard permettant de garder l'amplitude du signal. De plus on observe que les contours de l'image sont très dégradé par rapport à l'image souhaité et par rapport aux seuillage doux. Pour le seuillage doux l'image est plus régulière car la reconstruction est plus « continue » et l'aperçu visuel est de meilleur qualité même si l'on voit toujours l'effet du bruit. Le fait de retirer la valeur du seuil à chaque coefficient d'ondelettes supérieur au seuil permet de réduire le nombre de discontinuité dans l'image reconstruite. Pour les deux images on prend en considération que la valeur de sigma est relativement élevé d'où le fait qu'une reconstruction ne permettant pas d'éliminer complètement les effets du bruit.

On cherche maintenant le Threshold optimal pour le soft et le hard Thresholding, pour ce faire on applique un seuil relativement faible que l'on incrémente. Ensuite on calcul le rapport signal à bruit (SNR) afin d'évaluer le seuil optimal.

Le SNR étant défini par : $SNR(I_s/I_b) = 10^{-\log(\frac{signal}{bruit})}$

Avec le signal étant l'image de référence et le bruit étant l'image obtenue après traitement. On obtient alors le graphique suivant :



Pour le seuil on choisi : $T = 0.01 * \max(W1) * nIt$

Où $W1$ est la transformée en ondelette associée à notre image et donc on commence avec un seuil égal à 1% du coefficient maximum de la transformée. On itère alors en incrémentant nIt pour obtenir un éventail de seuil et calculé les différents SNR résultant.

Pour le soft Threshold on obtient $T_{\text{soft}} = 20.56$ ce qui correspond à 1% du coefficient maximum de la transformée en ondelette (= 2056.3) et pour le hard Threshold on obtient $T_{\text{hard}} = 53.46$ correspondant à 2.6% du maximum.

Même si avec le soft Threshold, on obtient au final une valeur de SNR légèrement plus élevée que pour le hard cette faible différence montre que le SNR seul ne suffit pas à déterminer la qualité de reconstruction de l'image mais donne une première optimisation.

Dans la littérature pour le soft Thresholding la valeur de seuil optimal est déterminé par la formule suivante :

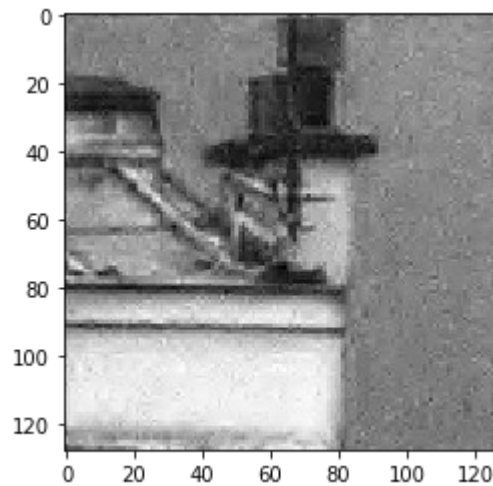
$$S = \sigma \sqrt{2 \log N}$$

Où N est le nombre d'élément du signal et σ est l'écart-type du signal donc la racine de la variance. Ce seuil permet de limiter le risque d'écart au signal.

Avec une variance égal à 20 et notre image étant de 128 par 128 pixel on obtient :

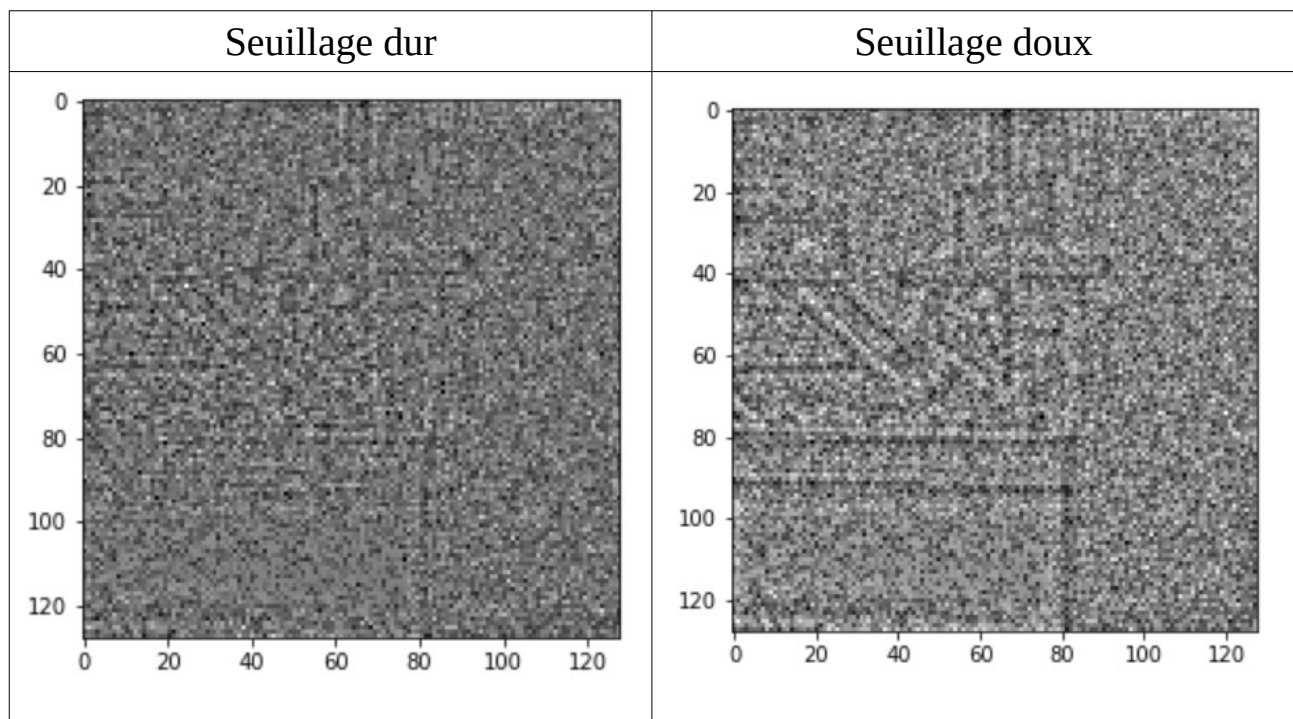
$$S = 19.7 \text{ ce qui se rapproche de notre résultat étant } S = 20.56.$$

En utilisant ce seuil optimal pour un soft Threshold on récupère l'image suivante :



Avec un SNR de 21.48 ce qui fait une différence de 0.13% avec notre résultat en partant des courbes tracées, ce qui montre que le seuil universelle permet de s'approcher du meilleur SNR possible et va être adapté à chaque image.

Avec ce résultat on peut regarder l'allure du bruit soustrait à l'image en prenant l'image bruitée auquel on vient soustraire le résultats débruité. On obtient alors :



On remarque alors que dans le bruit éliminé apparaît une partie du signal et que pour le seuillage dur la moyenne du bruit est de 4.7 contre 2.5 pour le seuillage doux. Le signal soustrait n'est donc pas tout à fait le bruit blanc que l'on a introduit car comporte de l'information hautes fréquence qui explique la perte de détail au niveau des contours de l'image reconstruite.

3. Estimation de la variance du bruit

Donoho et Johnstone propose d'estimer σ en se basant sur les coefficients empiriques des ondelettes. Afin d'estimer la variance du bruit on peut tout à fait partir de notre transformée en ondelettes et s'intéresser au niveau 1 de l'ondelette à J-1. En effet à ce niveau on a donc décomposé l'information basse fréquence et haute fréquence dans les différents cadrants de la transformée. En prenant les coefficients des cadrants hautes fréquences on peut alors estimer la variance du bruit. Ainsi Donoho et al. (1995) propose d'estimer la variance à l'aide de la formule :

$$\sigma_{estim} = \frac{M}{0.6745}$$

Où M correspond à la médiane des différents cadrants où il y a l'information haute fréquence.

En calculant la variance par cette méthode on obtient : $\sigma_{estimée} = 21.1$

Ce qui fait une erreur d'estimation de l'ordre de 5.5% qui est plutôt raisonnable et apporte ainsi une meilleure estimation de la variance.

4. Critiques

→ **Points positifs :**

L'intérêt du débruitage par ondelette est que la complexité algorithmique de ce débruitage dépend de la complexité de la transformée en ondelettes qui est en $N \log(N)$ ce qui apporte donc un algorithme rapide.

De plus pour effectuer des mesures de grandeur physiques dans une image, les coefficients d'ondelettes sont très représentatifs de ces grandeurs. Ainsi pour débruité l'image et pour obtenir des mesures qui se passe du bruit les ondelettes sont bien adaptés. En effet l'information du signal est contenue majoritairement dans le signal basse fréquence et pour retirer le bruit il suffit de seuiller les coefficients de la transformée en ondelette et de mettre à 0 le bruit.

→ **Points négatifs :**

Pour obtenir une image débruitée de bonne qualité "visuel" le débruitage par ondelette n'est pas forcément le plus adapté car fait apparaître des artefacts avec un seuil dur.

Avec un seuil doux comme avec un seuil dur on perd de l'information haute fréquence ce qui a pour conséquence de "flouté" l'image car on perd des détails.

De plus la transformée en ondelettes n'est pas invariante par translation.

III. NL Means

1. Analyse théorique

Pour cette recherche on s'est principalement basé sur les transparents et l'article de Jean-Michel Morel et al. Le but est de chercher dans l'image pour un pixel donnée un patch de 9 pixels ressemblant à ce pixel et d'appliquer une moyenne de ce patch aux niveaux du pixel donné. Si 9 pixels sont moyennés alors la variance du bruit est environ divisée par 3. Le patch n'étant pas forcément voisin direct du pixel. Il faut alors regarder une plus grande portion de l'image et regarder par exemple les patterns périodiques, les elongations (lignes). Le but est alors de créer un filtre moyens non local données par :

$$NLu(p) = \frac{1}{C(p)} \int f(d(B(p), B(q))) u(q) dq,$$

Avec :

- $d(B(p), B(q))$ la distance euclidienne entre les patchs centrés en p et en q ;
- f une fonction décroissante ;
- $C(p)$ un facteur de normalisation.

De plus avec $d(B(p), B(q))$ chaque pixel de $B(p)$ à la même importance et le poids $f(d(B(p), B(q)))$ peut être utilisé pour débruité l'ensemble des pixels du patch $B(p)$ on peut donc implémenter le débruitage niveau pixel ou niveau patch.

A - Niveau pixel

Le débruitage d'une image en couleur u et d'un certain pixel p :

$$\hat{u}_i(p) = \frac{1}{C(p)} \sum_{q \in B(p,r)} u_i(q) w(p, q), \quad C(p) = \sum_{q \in B(p,r)} w(p, q),$$

Avec :

- $B(p, r)$ les patchs centrés en p et de taille $(2r + 1)^2$ centrés sur le pixel p ;
- $u_i(q)$ les canaux de l'image (r,g,b) ;
- $w(p, q)$ le poids qui dépend de la distance euclidienne au carrée : $d^2 = d^2(B(p, f), B(q, f))$

$$d^2(B(p, f), B(q, f)) = \frac{1}{3(2f + 1)^2} \sum_{i=1}^3 \sum_{j \in B(0,f)} (u_i(p + j) - u_i(q + j))^2.$$

Le poids $w(p, q)$ permet la restauration de chaque pixel avec une moyenne des pixels ressemblant et est donnée par :

$$w(p, q) = e^{-\frac{\max(d^2 - 2\sigma^2, 0.0)}{h^2}}$$

Avec :

- σ l'écart-type du bruit ;
- h un paramètre de filtrage dépendant de σ , $h = k\sigma$ où k est décroissant en fonction de la taille du patch;

Les patchs ayant une distance carré plus petites que $2\sigma^2$, sont alors évalués avec un poids de 1 et les patchs avec une grande distance décroît très rapidement.

Le poids de la référence pour le pixel p est quant à lui fixé avec le poids maximum trouvé dans le voisinage $B(p, r)$.

B - Niveau patch

Le débruitage d'une image en couleur u et d'un certain patch $B(p, f)$ centré en p et de taille $(2f + 1)^2$:

$$\hat{B}_i = \frac{1}{C} \sum_{Q=Q(q,f) \in B(p,r)} u_i(Q) w(B, Q), \quad C = \sum_{Q=Q(q,f) \in B(p,r)} w(B, Q)$$

Avec :

- $B(p, r)$ les patchs centrés en p et de taille $(2r + 1)^2$ centrés sur le pixel p ;
- $u_i(q)$ les canaux de l'image (r,g,b) ;
- $w(B(p), B(q))$ le poids entre deux patchs en fonction de la distance qui les séparent, même implémentation que pour le niveau de pixel.

On alors $N^2 = (2f + 1)^2$ estimé possible pour chaque pixel, et on calcule l'image débruité avec :

$$\hat{u}_i(p) = \frac{1}{N^2} \sum_{Q=Q(q,f) | q \in B(p,f)} \hat{Q}_i(p)$$

L'ensemble des paramètres utilisés par la méthode de patch, pour différentes valeurs de σ :

Gray				Color			
σ	Comp. Patch	Res. Block	h	σ	Comp. Patch	Res. Block	h
$0 < \sigma \leq 15$	3×3	21×21	0.40σ	$0 < \sigma \leq 25$	3×3	21×21	0.55σ
$15 < \sigma \leq 30$	5×5	21×21	0.40σ	$25 < \sigma \leq 55$	5×5	35×35	0.40σ
$30 < \sigma \leq 45$	7×7	35×35	0.35σ	$55 < \sigma \leq 100$	7×7	35×35	0.35σ
$45 < \sigma \leq 75$	9×9	35×35	0.35σ				
$75 < \sigma \leq 100$	11×11	35×35	0.30σ				

C – Avantage et inconvénient des deux niveaux

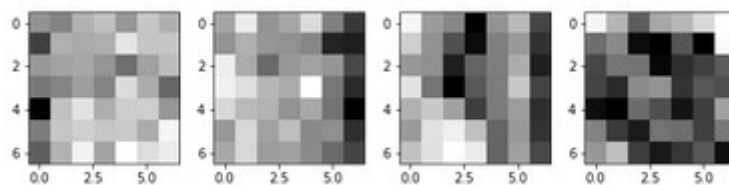
Il y a du bon et du mauvais dans les deux niveaux.

La méthode du patch permet d'améliorer le PSNR car le bruit est plus atténué, cela permet aussi de réduire les bruits parasite à proximité des bords.

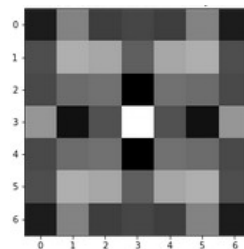
Par contre la qualité globale en terme de conservation des détails n'est pas améliorée par cette méthode.

2. Constatations numériques

Dans cette partie afin de pouvoir comparer les méthodes on utilise la mêmes image que pour la transformée en ondelette et on applique le même bruit blanc Gaussien de $\sigma = 20$. Pour commencer il faut donc préparer les patchs associés à chaque pixel. Donc pour chaque pixel on lui associe un patch de taille $(2r+1)(2r+1)$ avec r pris à 3. On obtient alors pour chaque pixel des patchs tel que :



Et au niveau des bords :



Pour éviter les effets de bords on réalise en fait un miroir des patches centrés sur les bords de l'image.

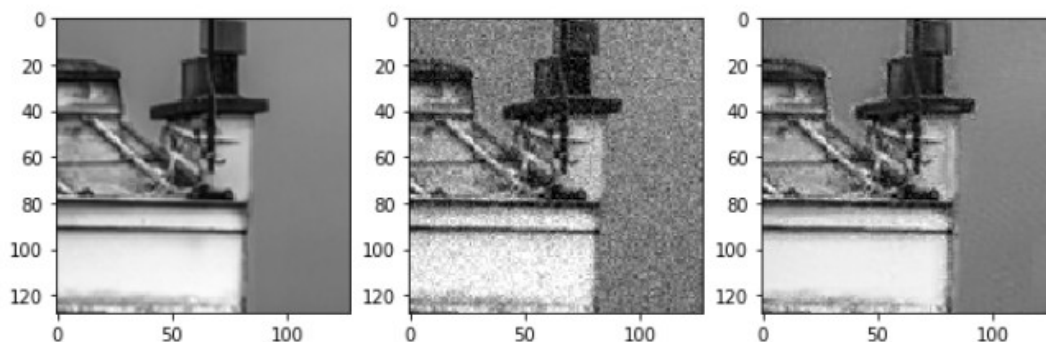
Une fois que l'on a l'ensemble des patches on peut utiliser une stratégie de PCA (Principal Component Analysis) sur un patch. Le but est alors de calculer la décomposition en valeur singulière (SVD) afin de pouvoir ranger par ordre croissant les « eigenvalues » correspondant aux composantes de l'image. En rangeant par ordre croissant les « eigenvalues » on vient donc faire l'analyse en composante principale car plus une « eigenvalue » est grande plus elle est importante. Pour la suite on vient donc réduire le nombre de composant d'un patch de $(2f+1)^2$ à une dimension N fixé, par exemple 15 ou 25 composantes. On vient alors travailler sur les N eigenvectors correspondant aux N eigenvalues afin de créer des nouveaux patches avec la dimension souhaité.

La méthode des NLMeans nécessite normalement pour chaque pixel p de parcourir l'ensemble de l'image et des pixels q associées. Par soucis de temps d'exécution il est préférable de travaillé sur des patches centrée en p de plus petite taille que l'image, ce sont les patches de tailles $(2r+1)$. Le choix de r va alors dépendre de la valeur de σ du bruit. Dans notre cas on peut ce rapporter à la table donné à la fin de l'article de Jean-Michel Morel et al. et pour un $\sigma = 20$ on doit prendre $r = 10$ et $f = 2$. Dans l'implémentation nous avons pris $r = 10$ et $f = 3$ et 25 dimensions pour le PCA qui nous donne alors un résultat plus satisfaisant.

Quelque choix de f et r différent :

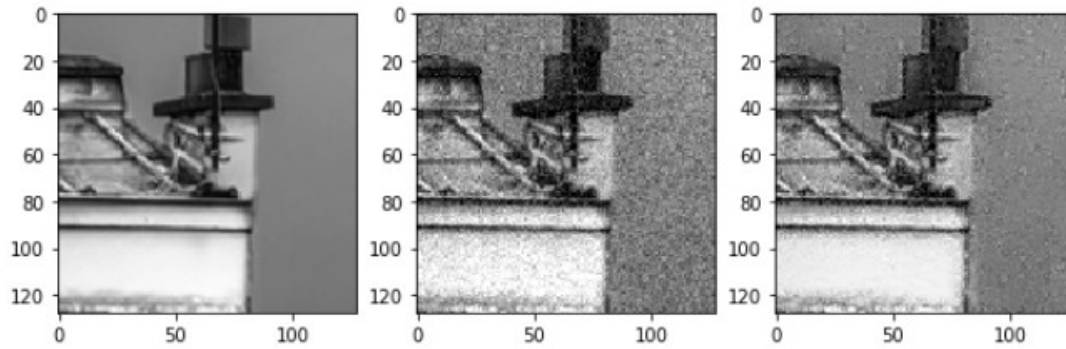
Meilleur SNR, $k=0.4$:

```
tau= 8.0 f= 3 r= 10 sigma= 20.0 snr(image,imgdenoised) = 22.461652478918058
```



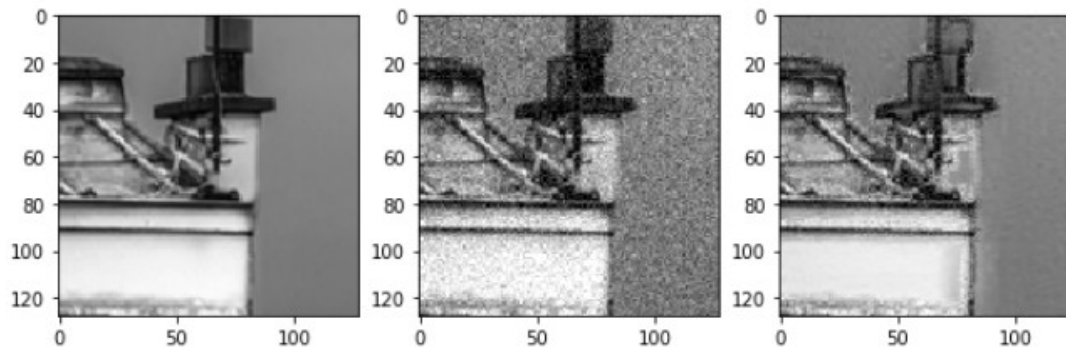
Choix de f et r en fonction de la table, $k = 0.4$:

tau= 8.0 f= 2 r= 10 sigma= 20.0 snr(image,imgdenoised) = 19.934422253752015



Test pour des valeurs de la table pour des σ compris entre 30 et 45 et $k = 0.35$:

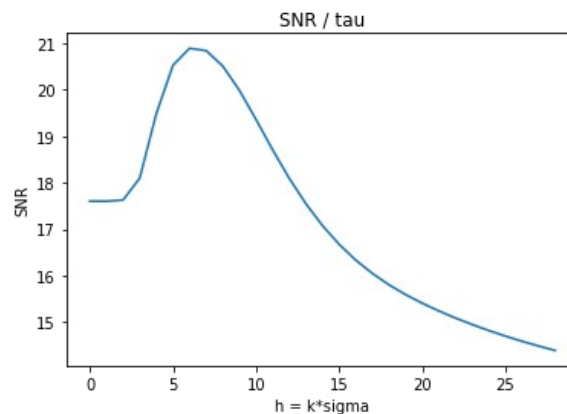
tau= 7.0 f= 3 r= 17 sigma= 20.0 snr(image,imgdenoised) = 19.88711245936464



Les valeurs fournis donne donc de bon résultats pour généraliser leur travail sur un ensemble de bruit différent, mais donc dans notre cas on obtient un SNR de meilleur qualité pour $f = 3$ et $r = 10$ pour un $\sigma = 20$ ce qui ne correspond pas à un des résultats de la table. Et niveau visuel ce résultat nous semble aussi plus satisfaisant que les deux autres.

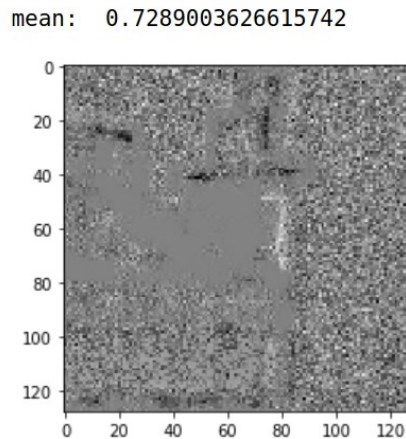
Pour $f = 2$ et $r = 10$ la taille des patches semble trop petites et l'image reste très bruitée, tandis que pour $f = 3$ et $r = 17$ la fenêtre de recherche est plus importante et l'on observe une perte d'information. Pour ce second cas cela montre que l'application des NLMeans reste tout de même un algorithme semi-local et que la taille de r va dépendre de la valeur de σ . Plus σ est grand plus le bruit est important et plus il faut récupérer de l'information "loin" du pixel p considéré.

On réalise ci-dessous la courbe de SNR en fonction du choix de $h = k * \sigma$:



Pour $\sigma = 20$ on retombe effectivement sur les valeurs du tableau, en effet le meilleur SNR est alors obtenue pour $h = 0.4 * \sigma$ comme indiqué par Jean-Michel Morel et al.

Comme pour les ondelettes on peut regarder alors le bruit éliminé entre l'image débruité et l'image bruité on obtient :



Ici on peut voir qu'une partie du signal souhaité reste quand même éliminée mais en moyenne on se rapproche plus de 0 que pour les ondelettes. De plus le bruit ainsi visualisé se rapproche plus d'un bruit blanc que le bruit d'ondelettes. Et on peut voir que sur l'image du bruit les contours sont moins affectés que sur l'image de bruit obtenu avec les ondelettes. On remarque tout de même que dans certaines régions homogènes de l'image initiale le bruit est lui aussi homogène ce qui montre que peut-être du bruit a été éliminé dans ces régions. Cela nous montre au moins pour cette image que le débruitage par NL-means permet d'éliminer un bruit plus en relation avec le bruit introduit dans l'image, ici le bruit blanc.

3. Estimation de la variance du bruit

En pratique la variance du bruit n'est pas connue dans le signal. Pour la calculer on décompose l'image en un ensemble de patchs de taille r . Chaque pixel recevra alors un patch de taille $(r+1)*(r+1)$. On parcourt alors chaque patch ainsi créé et on calcule la variance de chaque patch p :

$$Var(p) = \mathbb{E}(p - \mathbb{E}(p))^2 = \mathbb{E}(p^2) - \mathbb{E}(p)^2$$

On estime alors la variance du bruit comme étant représentée par la moyenne des 10% des variances des patchs calculés. En effet comme expliqué un peu plus tôt le bruit blanc se répartit équitablement sur l'ensemble du signal mais son impact est alors plus important sur les hautes fréquences et donc sur les petits coefficients dans l'image.

Cette estimation nous donne alors : $\sigma_{estimée} = 25.5$ contre $\sigma_{attendu} = 20$.

Il y a 27.5% d'erreur sur l'estimation.

L'erreur vient du fait que le choix des 10% n'est pas forcément adapté à toutes les images mais qu'en moyenne cela permet d'obtenir une estimation proche de la variance du bruit tout de même. Ici l'estimation n'est pas optimale et l'erreur est assez importante, l'idée derrière nous semblait plutôt intéressante mais on se retrouve avec une estimation moins intéressante que celle calculée pour les ondelettes.

4. Critiques

→ Points positives :

Le bruit soustrait par méthode des NLMeans est généralement proche d'un bruit blanc ce qui permet d'avoir un meilleur denoising.

Le fait de travailler de façon non local permet d'avoir un meilleur moyennage que par méthode classique et présente alors un meilleur denoising.

Il n'y a pas d'artefacts dans l'image débruitée, ce qui permet d'obtenir un meilleur résultat visuel.

→ Points négatives :

La complexité des NLMeans est très importante car elle est quadratique:

Pour une image de tailles $N \times M$, avec des patches (paramètre f) de taille 7×7 et une fenêtre de recherche (paramètre r) de taille 21×21 , étant les paramètres les plus communément utilisés, la complexité associée est :

$$\text{complexité} = (2f+1)^2 \cdot (2r+1)^2 \cdot N \cdot M = 7^2 \cdot 21^2 \cdot N \cdot M$$

Il existe cependant des méthodes pour optimiser cette complexité par exemple Laurent Condat propose d'utiliser de la convolution dans les étapes de NLMeans pour améliorer la vitesse d'exécution dans son article : [A Simple Trick to Speed Up the Non-Local Means](#).

Pour le débruitage l'article de Simon Postec et al. « Non-Local means est un algorithme de débruitage local » suggère après étude que pour obtenir de meilleure performance de débruitage il faut réduire la taille des patches de recherche. En plus du caractère coûteux en tant de calcul des NLMeans ils montrent que les meilleures performances en moyenne de cet algorithme pour du débruitage restent dans un rayonnement local du pixel considéré.

IV. Conclusion

Dans ce papier nous avons donc mis en place deux méthodes de débruitage que sont le débruitage par NL-means et un débruitage par Ondelettes.

Le débruitage par Ondelettes utilise le fait que les coefficients faibles présent dans les ondelettes représente le bruit et on peut donc venir seuillez ces coefficients en perdant moins d'information hautes fréquence que par l'application d'une passe-bas directement sur l'image. Un des principales avantages de cette méthodes vient du fait de sa complexité en $N\log(N)$ lié directement à la complexité de la transformé en ondelettes. De plus la transformé en ondelettes préservant l'énergie le résultats du débruitage est alors exploitable pour faire des mesures sur l'image et donne alors des résultats très satisfaisant sur des images présentant de forte discontinuité. Par exemple sur des images de radar, satellites qui sont la plus part du temps très bruitées.

L'algorithme des NL-means quant à lui semble fournir des résultats plus intéressant que l'algorithmes par Ondelettes. On obtient un meilleur SNR ainsi qu'une meilleur reconstruction visuel de notre image. Le fait d'utilisé des patchs de ressemblance dans l'image apporte une solution qui semble assez efficace. Chaque pixel étant moyenné par un ensemble de pixel qui lui ressemble cela convient aux problèmes de denoising. Par rapport à des méthodes classique de moyennage, regardé l'image de façon non local apporte une meilleur correction et le bruit sortie d'un débruitage par NL-means ce rapproche plus d'un bruit blanc. Un des problèmes majeur de cette méthode vient de sa complexité qui est quadratique (en N^2) et que de ce fait il faut trouver des stratégies pour réduire celle-ci. De base la méthode travaille avec un patchs pour réduire le nombre de calcule, de plus dans l'implémentation proposé on utilise une méthode de PCA pour réduire la complexité. Des méthodes comme l'introduction de convolution que propose Laurent Condat sont sûrement mieux adapté pour répondre aux problèmes de complexité.

Afin de débruité une image on retient alors que selon l'objectif souhaité la méthode employé va différé et va dépendre de la qualité de reconstruction mais aussi du temps d'exécution. Ainsi, parmi les deux solutions présentée si l'objectif est d'avoir un bon résultats visuel, les NLmeans seront plus adapté. Mais si l'on souhaite effectué un grand nombre de calcul et de travaille sur une série d'image, aux prix d'une légère erreur, la méthode de denoising par Ondelettes sera mieux adaptés du fait de sa rapidité d'exécution et de résultats plus que convenable aux niveaux des propriétés d'une image.

V. Bibliographie

→ François Chaplais. ENSMP [En ligne]. [Consultée le 27 janvier 2020]. Disponible sur : <http://cas.ensmp.fr/~chaplais/fr/Ondelettes/Ondelettes/DebruitageOndelettes.html>

→ ENST [En ligne]. [Consultée le 27 janvier 2020]. Disponible sur : <http://www.tsi.enst.fr/pages/enseignement/ressources/mti/donoho/Seuillage/SeuilCoeff.htm>

→ Wikipédia [En ligne]. [Consultée le 27 janvier 2020]. Disponible sur : https://fr.wikipedia.org/wiki/D%C3%A9bruitage_par_patches

→ Antoni Buades, Bartomeu Coll, Jean-Michel More. IPOL, Image Processing On Line [En ligne]. CC-BY-NC-SA [consultée le 27 janvier 2020]. Disponible sur : http://www.ipol.im/pub/art/2011/bcm_nlm/article.pdf

→ Laurent Condat. HAL archives ouvertes [En ligne]. A Simple Trick to Speed Up the Non-Local Means. 2010. fffhal-00512801v1f [consultée le 13 février 2020]. Disponible sur : <https://hal.archives-ouvertes.fr/file/index/docid/512801/filename/hal.pdf>

→ Wikipédia [En ligne]. [Consultée le 27 février 2020]. Disponible sur : https://en.wikipedia.org/wiki/Signal-to-noise_ratio

→ Wikipédia [En ligne]. [Consultée le 27 février 2020]. Disponible sur : https://fr.wikipedia.org/wiki/D%C3%A9composition_en_valeurs_singuli%C3%A8res

→ Wikipédia [En ligne]. [Consultée le 27 février 2020]. Disponible sur : https://fr.wikipedia.org/wiki/Analyse_en_composantes_principales

VI. Glossaire

→ **Non Local means** (débruitage par patches) est une technique de débruitage d'image utilisant l'algorithme de réduction du bruit numérique.

Contrairement aux filtres habituels qui réalisent une moyenne des valeurs du groupe de pixels localisés autour d'un pixel cible afin de réduire le bruit, le filtre "non-local means" réalise une moyenne de la totalité des valeurs des pixels contenus dans l'image, pondérées en fonction de leur similarité avec le pixel cible. Le résultat d'un tel filtrage permet d'amoindrir la perte de détails au sein de l'image, comparé aux filtres réalisant des moyennes localement.

→ **SNR** (Signal to Noise Ratio) est un indicateur de la qualité de la transmission d'une information. C'est le rapport des puissances entre :

- le signal d'amplitude maximale pour laquelle la distorsion à la sortie reste inférieure à une valeur limite ;
- le bruit de fond, information non significative correspondant en général au signal présent à la sortie du dispositif en l'absence d'un signal à l'entrée.

Il s'exprime généralement en décibels (dB)

→ **PSNR** (sigle de Peak Signal to Noise Ratio) est une mesure de distorsion utilisée en image numérique, tout particulièrement en compression d'image. Elle permet de quantifier la performance des codeurs en mesurant la qualité de reconstruction de l'image compressée par rapport à l'image originale.

→ **Hard Thresholding** est un seuillage dur (le plus intuitif).

→ **Soft Thresholding** est un seuillage doux.

→ **PCA** (Principal Component Analysis) est l'analyse en composantes principales, il s'agit d'une stratégie qui consiste à transformer des variables liées entre elles en nouvelles variables décorréliées les unes des autres. Ces nouvelles variables sont nommées « composantes principales », ou axes principaux. Elle permet au praticien de réduire le nombre de variables et de rendre l'information moins redondante.

→ **SVD** (Singular Value Decomposition) est une décomposition en valeurs singulières d'une matrice qui est un outil important de factorisation des matrices rectangulaires.

→ Eigen Values / EigenVector

VII. Annexe