Dé-bruitage d'images : BM3D et “Non-Local Bayes”

Théo Cornille - Luc Gibaud - Cédric des Lauriers

Décembre 2018

Introduction

L'objectif de ce projet est d'étudier différents algorithmes utilisés pour le débruitage d'images. En particulier, nous avons retenu notre attention sur deux algorithmes : BM3D et "Non-Local Bayes". Ces deux algorithmes sont en effet très similaires dans leur implémentation et également au niveau des résultats obtenus. Cependant, nous allons voir ce qui les distingue.

Travail effectué

Nous avons implémenté BM3D et NL-Bayes en python. En particulier, nous avons pour cela utilisé les librairies : numpy, random et cv2.

Présentation des deux algorithmes

Les 2 algorithmes sont très similaires par leur composition. Ils se décomposent en deux étapes. La première étape va extraire une estimation de l’image bruitée et la deuxième étape va utiliser l’estimation obtenu à la première étape comme oracle pour débruiter l’image initiale.

Chacune des deux parties contiennent les mêmes phases :

Une phase de grouping qui va chercher à trouver et extraire les patchs similaires par rapport à un patch centrée et qui seront rassemblés pour former un bloc 3D.

Une phase de Collaborative Filtering qui permet de filtrer les blocs d’une certaine manière afin d’obtenir des caractéristiques pour le débruitage. C’est principalement au niveau de cette phase que les algorithmes vont se distinguer. Nous les présenterons dans la suite.

Une phase d’aggregation qui permet de repositionner les blocs 3D à leur place d’origine et d’effectuer un moyennage des informations redondantes qui peuvent exister dans les blocs 3D.

BM3D

BM3D signifie Block Matching and 3D filtering.

L'algorithme BM3D est une fusion de l'algorithme NL-means et d'un seuillage DCT et est très efficace pour le débruitage d'images.

La différence avec le débruitage DCT est l'utilisation de patchs similaires rassemblés en bloc 3D puis filtrés par une transformation 3D d'où le nom de « collaborative filtering ».

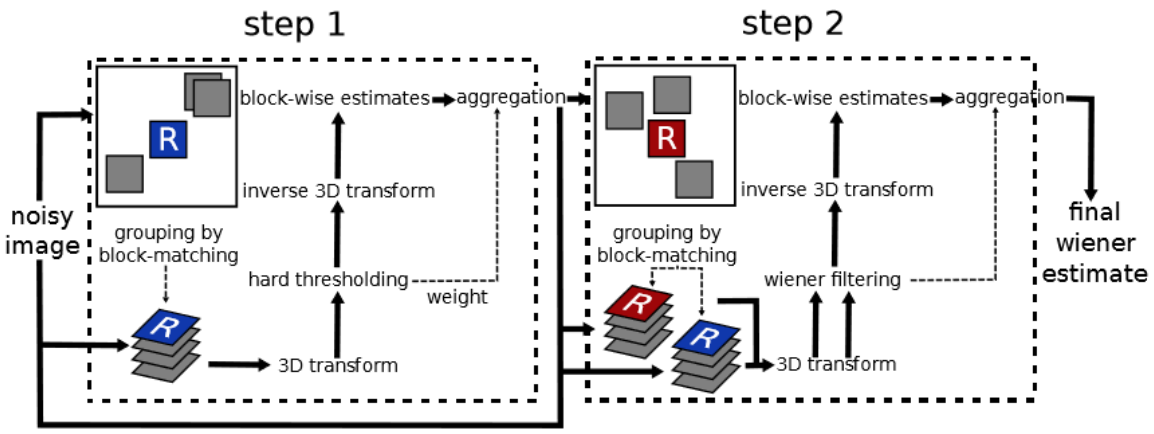


Figure 1: Scheme of the BM3D algorithm.

Voici l'algorithme BM3D sous sa forme détaillée :

Step 1 : Premier Débruitage de l'image. P correspond au patch de bruit de référence

Grouping : Construction du bloc 3D *P*(P) à partir des patchs similaires à P obtenus à partir d'un seuil de similarité sur la distance entre patchs.

Collaborative Filtering : Les blocs 3D subissent une transformation linéaire 3D puis un seuillage dur des coefficients et enfin une transformation 3D inverse.

Aggregation : Une première estimation de l'image débruité est obtenue en faisant une agrégation pondérée de toutes les estimations obtenuEs à l'étape précédente pour chaque pixel.

L'estimation basique est dénotée u^{basic}

Step 2 : Second Débruitage utilisant l'estimation du 1er step comme « oracle ».

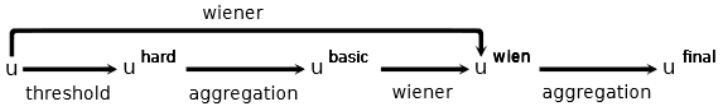
Grouping : La distance entre les patchs est calculée sur l'estimation basique. Deux blocs 3D sont formés :

- *Pbasic*(Pbasic) qui utilisent les patchs similaires de Pbasic (de l'image ubasic) à partir du patch de référence Pbasic pour former le bloc 3D
- *Pbasic*(P) qui utilise les patchs similaires de Pbasic à partir du patch de référence P (issue de l'image u).

Collaborative Filtering : Une transformation 3D est appliquée sur les blocs 3D, puis un filtrage à partir de Wiener sur le groupe *Pbasic*(P) et utilisant les coefficients obtenus du groupe *Pbasic*(Pbasic), et enfin une transformation 3D inverse.

Aggregation : Une estimation finale de l'image débruitée est obtenue en utilisant une agrégation pondérée de chaque estimation obtenue pour chaque pixel. L'estimation finale est dénotée u^{final}.

On voit donc que les phases de Collaborative Filtering des steps 1 et 2 se distinguent par l'utilisation d'un seuillage dur des coefficients pour le step 1 tandis que pour le step 2, un filtre de Wiener est appliqué sur l'image préalablement débruitée dans le step 1.



Pseudocode

Step 1 :

Algorithm 11 BM3D first iteration algorithm for grey images.

Input: noisy image $\tilde{u}$, σ, noise standard deviation.
Output: output basic estimation $\hat{u}_1$ of the denoised image.

Set parameter $\kappa \times \kappa = 8 \times 8$: dimension of patches.
Set parameter $\lambda \times \lambda = 39 \times 39$: size of search zone in which similar patches are searched.
Set parameter $N_{max} = 16$: maximum number of similar patches retained during the grouping part.
Set parameter $s = 3$: step in both rows and columns between two reference patches.
Set parameter $\lambda_{3D} = 2.7$: coefficient used for the hard thresholding.
Set parameter $\tau = 2500$ (if $\sigma > 40, \tau = 5000$): threshold used to determine similarity between patches.

for each pixel **i**, with a step s in rows and columns **do**

Select a square reference patch $\tilde{P}$ around **i** of size $\kappa \times \kappa$.

Look for square patches $\tilde{Q}$ in a square neighborhood of **i** of size $\lambda \times \lambda$ having a distance to $\tilde{P}$ lower than τ.

if there are more than N_{max} similar patches **then**
keep only the N_{max} closest similar patches to $\tilde{P}$ according to their Euclidean distance.
else
keep 2^p patches, where p is the largest integer such that 2^p is smaller than the number of similar patches
end if

A 3D group $\mathcal{P}(\tilde{P})$ is built with those similar patches.

A bi-orthogonal spline wavelet (Bior 1.5) is applied on every patch contained in $\mathcal{P}(\tilde{P})$.

A Walsh-Hadamard transform is then applied along the third dimension of the 3D group $\mathcal{P}(\tilde{P})$.

A hard thresholding with threshold $\lambda_{3D}\sigma$ is applied to $\mathcal{P}(\tilde{P})$. An associated weight $w_{\tilde{P}}$ is computed :

$$w_{\tilde{P}} = \begin{cases} (N_{\tilde{P}})^{-1} & N_{\tilde{P}} \geq 1 \\ 1 & N_{\tilde{P}} = 0 \end{cases}$$

where $N_{\tilde{P}}$ is the number of retained (non-zero) coefficients.

The estimate $\hat{u}_1^{\tilde{Q},\tilde{P}}$ for each pixel **i** in similar patches $\tilde{Q}$ of the 3D group $\mathcal{P}(\tilde{P})$ is then obtained by applying the inverse of the Walsh-Hadamard transform along the third dimension, followed by the inverse of the bi-orthogonal spline wavelet on every patches of the 3D group.

end for

for each pixel **i** **do**

Aggregation: recover the denoised value at **i** by averaging all estimates of all patches $\tilde{Q}$ in all 3D groups $\mathcal{P}(\tilde{P})$ containing **i**, the weights being given by the $w_{\tilde{P}}$.

end for

Step 2 :

Algorithm 12 BM3D second iteration algorithm for grey images.

Input: noisy image $\tilde{u}$, σ, noise standard deviation.
Input: basic estimation $\hat{u}_1$ obtained at the first step.
Output: final denoised image $\hat{u}$.

Set parameter $\kappa \times \kappa = 8 \times 8$ (up to 12 for high noise level): dimension of patches.
Set parameter $\lambda \times \lambda = 39 \times 39$: size of search zone in which similar patches are searched.
Set parameter $N_{max} = 32$: maximum number of similar patches retained during the grouping part.
Set parameter $s = 3$: step in both rows and columns between two reference patches.
Set parameter $\tau = 400$ (if $\sigma > 40, \tau = 3500$): threshold used to determinate similarity between patches.

for each pixel **i**, with a step s in rows and columns **do**

Take the square reference patches $\tilde{P}$ and $\hat{P}_1$ centered at **i**, of size $\kappa \times \kappa$ in the initial and basic estimation images.

Look for square patches $\hat{Q}_1$ in a square neighborhood of **i** of size $zsize \times zsize$ having a distance lower than τ in the basic estimate image $\hat{u}_1$.

if there are more than N_{max} similar patches **then**

keep only the N_{max} closest similar patches to $\hat{P}_1$ according to their Euclidean distance.

else

keep 2^p patches, where p is the largest integer such that 2^p is smaller than the number of similar patches

end if

Two 3D groups $\mathcal{P}(\tilde{P})$ and $\mathcal{P}(\hat{P}_1)$ are built with those similar patches, one from the noisy image $\tilde{u}$ and one from the basic estimate image $\hat{u}_1$.

A 2D DCT is applied on every patch contained in $\mathcal{P}(\tilde{P})$ and $\mathcal{P}(\hat{P}_1)$.

A Walsh-Hadamard transform is then applied along the third dimension of $\mathcal{P}(\tilde{P})$ and $\mathcal{P}(\hat{P}_1)$.

Denoting by τ_{3D} the 3D transform (2D DCT followed by the Walsh-Hadamard transform) applied on the 3D group, compute the Wiener coefficient $\omega_{\tilde{P}} = \frac{|\tau_{3D}(\mathcal{P}(\hat{P}_1)|^2}{|\tau_{3D}(\mathcal{P}(\hat{P}_1)|^2+\sigma^2}$.

The Wiener collaborative filtering of $\mathcal{P}(\tilde{P})$ is realized as the element-by-element multiplication of the 3D transform of the noisy image $\tau_{3D}(\mathcal{P}(\tilde{P}))$ with the Wiener coefficients $\omega_{\tilde{P}}$.

An associated weight $w_{\tilde{P}}$ is computed :

$$w_{\tilde{P}} = \begin{cases} (\|\omega_{\tilde{P}}\|_2)^{-2} & \|\omega_{\tilde{P}}\|_2 > 0 \\ 1 & \|\omega_{\tilde{P}}\|_2 = 0 \end{cases}$$

The estimate $\hat{u}_2^{\tilde{Q},\tilde{P}}$ for each pixel $\mathbf{i}$ in similar patches $\tilde{Q}$ of the 3D group $\mathcal{P}(\tilde{P})$ is then obtained by applying the inverse of the 1D Walsh-Hadamard transform along the third dimension, followed by the inverse of the 2D DCT on every patch of the 3D group.

end for

for each pixel $\mathbf{i}$ **do**

Aggregation: Recover the denoised value $\hat{u}(\mathbf{i})$ at $\mathbf{i}$ by averaging all estimates of patches $\tilde{Q}$ in all 3D groups $\mathcal{P}(\tilde{P})$ containing $\mathbf{i}$, using the weights $\omega_{\tilde{P}}$.

end for

NL-BAYES

NL-Bayes est une variante améliorée de l'algorithme NL-means. Dans NL-means, chaque patch est remplacé par une moyenne pondérée des patchs les plus similaires présents au voisinage.

L'algorithme NL-Bayes permet d'améliorer NL-means en évaluant pour chaque groupe de patchs similaires un modèle de vecteur gaussien. Pour chaque patch on a donc une moyenne pondérée mais aussi une matrice de covariance qui estime la variabilité du groupe de patchs similaires. Cela permet de calculer l'estimation optimal (au sens de l'erreur quadratique moyenne bayésienne minimal) pour chaque patch bruité dans le groupe, par l'inversion de la matrice de covariance. Et ça grâce à la formule suivante :

$$\hat{P}_1 = \overline{\tilde{P}} + \left[\mathbf{C}_{\tilde{P}} - \sigma^2\mathbf{I}\right]\mathbf{C}_{\tilde{P}}^{-1}(\tilde{P} - \overline{\tilde{P}})$$

L'implémentation se fait en 2 itérations identiques mais la deuxième itération utilise l'image débruité de la 1ère itération pour mieux estimer la moyenne et la covariance du patch de modele gaussien.

Nous allons expliquer comment NL-Bayes fonctionne :

Tout d'abord, *P*(P) correspond au groupe de patchs Q similaires au patch P, obtenus avec un seuil de tolérance honorable pour pouvoir dire qu'ils sont une version bruitée de P. Par la loi des grands nombres, on a :

$$\mathbf{C}_{\tilde{P}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P}) - 1}\sum_{\tilde{Q}\in\mathcal{P}(\tilde{P})}\left(\tilde{Q} - \overline{\tilde{P}}\right)\left(\tilde{Q} - \overline{\tilde{P}}\right)^t, \qquad \overline{\tilde{P}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P})}\sum_{\tilde{Q}\in\mathcal{P}(\tilde{P})}\tilde{Q}.$$

Cependant, la sélection des patchs similaires lors de la 1ère itération n'est pas optimale et peut être améliorée lors d'une seconde estimation où la première estimation est utilisée comme oracle.

On peut donc recalculer Cp1 et la nouvelle estimation P1 à partir de l'image débruité lors de la 1ère itération (dénoté P1 dans les formules ci-dessous)

$$\mathbf{C}_{\hat{P}_1} \simeq \frac{1}{\#\mathcal{P}(\hat{P}_1) - 1} \sum_{\hat{Q}_1 \in \mathcal{P}(\hat{P}_1)} \left(\hat{Q}_1 - \overline{\hat{P}}^1\right)\left(\hat{Q}_1 - \overline{\hat{P}}^1\right)^t, \qquad \overline{\hat{P}}^1 \simeq \frac{1}{\#\mathcal{P}(\hat{P}_1)} \sum_{\hat{Q}_1 \in \mathcal{P}(\hat{P}_1)} \tilde{Q}.$$

Et ainsi calculer l'estimation finale :

$$\hat{P}_2 = \overline{\hat{P}}^1 + \mathbf{C}_{\hat{P}_1} \left[\mathbf{C}_{\hat{P}_1} + \sigma^2 \mathbf{I}\right]^{-1} (\tilde{P} - \overline{\hat{P}}^1)$$

Pseudo Code

Algorithm 5 Non local Bayes image denoising

Input: noisy image
Output: denoised image

for all patches $\tilde{P}$ of the noisy image **do**

Find a set $\mathcal{P}(\tilde{P})$ of patches $\tilde{Q}$ similar to $\tilde{P}$.

Compute the expectation $\overline{\tilde{P}}$ and covariance matrix $\mathbf{C}_{\tilde{P}}$ of these patches by

$$\mathbf{C}_{\tilde{P}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P}) - 1} \sum_{\tilde{Q} \in \mathcal{P}(\tilde{P})} \left(\tilde{Q} - \overline{\tilde{P}}\right)\left(\tilde{Q} - \overline{\tilde{P}}\right)^t, \qquad \overline{\tilde{P}} \simeq \frac{1}{\#\mathcal{P}(\tilde{P})} \sum_{\tilde{Q} \in \mathcal{P}(\tilde{P})} \tilde{Q}.$$

Obtain the first step estimation:

$$\hat{P}_1 = \overline{\tilde{P}} + \left[\mathbf{C}_{\tilde{P}} - \sigma^2 \mathbf{I}\right] \mathbf{C}_{\tilde{P}}^{-1} (\tilde{P} - \overline{\tilde{P}}).$$

end for
Obtain the pixel value of the basic estimate image $\hat{u}_1$ as an average of all values of all denoised patches $\hat{Q}_1$ which contain $\mathbf{i}$.
for all patches $\tilde{P}$ of the noisy image **do**

Find a new set $\mathcal{P}_1(\tilde{P})$ of noisy patches $\tilde{Q}$ similar to $\tilde{P}$ by comparing their denoised "oracular" versions Q_1 to P_1.

Compute the new expectation $\overline{\hat{P}}^1$ and covariance matrix $\mathbf{C}_{\hat{P}_1}$ of these patches:

$$\mathbf{C}_{\hat{P}_1} \simeq \frac{1}{\#\mathcal{P}(\hat{P}_1) - 1} \sum_{\hat{Q}_1 \in \mathcal{P}(\hat{P}_1)} \left(\hat{Q}_1 - \overline{\hat{P}}^1\right)\left(\hat{Q}_1 - \overline{\hat{P}}^1\right)^t, \qquad \overline{\hat{P}}^1 \simeq \frac{1}{\#\mathcal{P}(\hat{P}_1)} \sum_{\hat{Q}_1 \in \mathcal{P}(\hat{P}_1)} \tilde{Q}.$$

Obtain the second step patch estimate

$$\hat{P}_2 = \overline{\hat{P}}^1 + \mathbf{C}_{\hat{P}_1} \left[\mathbf{C}_{\hat{P}_1} + \sigma^2 \mathbf{I}\right]^{-1} (\tilde{P} - \overline{\hat{P}}^1).$$

end for
Obtain the pixel value of the denoised image $\hat{u}(\mathbf{i})$ as an average of all values of all denoised patches $\hat{Q}_2$ which contain $\mathbf{i}$.

Distinction avec BM3D :

Bien que NL-Bayes et BM3D soient très similaires, nous allons voir ce qui les différencient.

Tout d'abord, NL-Bayes se distingue de BM3D par sa phase de collaborative filtering.

On a pour NL-Bayes, l'utilisation du théorème de Bayes qui nous permet d'avoir la matrice de covariance et l'estimation de l'image débruité à partir de l'inversion de cette matrice.

Tandis que pour BM3D, c'est principalement l'utilisation du filtre de Wiener qui nous permet d'avoir cette estimation.

Il y aussi d'autres petites différences telles que :

- Le chevauchement des patchs qui est d'écart 3 pour BM3D tandis qu'il n'est que de 1 pour NL-Bayes
- L'utilisation des patchs qui correspond à la totalité pour NL-Bayes tandis que BM3D n'utilise que les N1 meilleurs.
- Ou encore un seuillage qui est fixe pour BM3D et adaptif pour NL-Bayes.

Partie expérimentale

Nos algorithmes désormais implémentés et fonctionnels, nous allons les tester et les comparer.

Pour les expériences, nous nous appuierons sur deux mesures :

- La Root Mean Square Error (RMSE)

$$RMSE = \sqrt{\frac{\sum_{x \in X}(u_R(x) - u_D(x))^2}{|X|}}$$

Avec u_R l'image de référence sans bruit et u_D l'image débruité.

Plus le RMSE est faible et meilleur est le débruitage.

- Le Peak Signal to Noise Ratio (PSNR) évalué en décibels (dB)

$$PSNR = 20log_{10}(\frac{255}{RMSE})$$

Plus le PSNR est grand, meilleur est le débruitage.

Pour tester ces deux algorithmes de débruitage nous allons appliquer avec une plus ou moins forte intensité un bruit gaussien avec 2 images différentes : Barbara.png et Boat.png (toutes deux en noir et blanc).

Barbara

Boat

Pour évaluer la qualité de débruitage de BM3D et NL BAYES nous calculerons le PSNR et le RMSE de l'image ayant reçu du bruit avec l'image originale, ainsi que le PSNR et RMSE de l'image ayant reçu du bruit, puis étant débruiter avec l'un de ces algorithmes, avec l'image originale. On pourra donc quantifier la qualité de ces derniers.

De même nous comparons le BM3D et le NL Bayes avec des techniques de bruitage dites plus simple telles que le filtre gaussien, médian et maximum en se fiant au PSNR obtenu avec ces dernières techniques.

Pour finir on appliquera le BM3D sur différents types de bruit tels que le rayleigh, speckle et d'autres (non exhaustif).

PSNR noised : PSNR(image_originale, image_bruitée)

RMSE noised : RMSE(image_originale, image_bruitée)

Algo = BM3D, NL BAYES, ou différents filtre (gaussien, median, maximum)

PSNR denoised : PSNR(image_originale, Algo[image_bruitée])

RMSE denoised : RMSE(image_originale, Algo[image_bruitée])

Résultats BM3D sur Barbara.png et Boat.png

Barbara

Sigma	PSNR noised	RMSE noised	PSNR denoised	RMSE denoised
2	43.9	2.7	30.8	54.1
5	34.8	21.6	30.4	59.4
10	28.4	93.2	29.49	73.0
20	22.3	379.7	27.8	107.1
30	18.9	841	26.5	145
40	16.4	1473	25.8	173
60	13.4	3000	24.3	240
80	11.5	4616	23.1	316
100	10.3	6133	21.8	432

Boat

Sigma	PSNR noised	RMSE noised	PSNR denoised	RMSE denoised
2	43.9	2.7	33.95	26.2
5	34.9	21.3	33.5	29.1
10	28.5	92.4	32.7	35.2
20	22.4	374.4	31	52.4
30	19	824	29.5	73.5
40	16.6	1415	28.4	94
60	13.5	2895	25.9	169
80	11.6	4506	23.9	270
100	10.26	6119	21.9	420.1

Résultat NL Bayes

Barbara

Sigma	PSNR noised	RMSE noised	PSNR denoised	RMSE denoised
2	43.78	2.7	34.0	25.6
5	34.8	21.6	33.8	26.9
10	28.4	94.2	33.0	32.7
20	22.3	382.8	30.5	57.9
30	18.9	840	28.5	92.4
40	16.5	1441	26.4	149
60	13.4	2968	24.3	243.7
80	11.5	4574	23.1	320
100	10.4	5971	21.9	417

Boat

Sigma	PSNR noised	RMSE noised	PSNR denoised	RMSE denoised
2	43.8	2.7	37.4	11.8
5	34.9	21.3	37	13.0
10	28.5	91.7	35.7	17.5
20	22.4	373.9	32.8	33.4
30	19.0	818	30.7	54.9
40	16.6	1419	28.7	87.7
60	13.5	2887	26.1	160.7
80	11.5	4556.9	23.7	273.9
100	10.3	6072	22	408

Remarque : Le bruit n'est pas tout à fait le même entre les expériences sur le BM3D et le NL Bayes pour plus de rigueur on aurait dû appliquer exactement le même bruit sur les images et débruiter sur les mêmes images bruitées au lieu de le faire séparément c'est une erreur de ma part (C.D.L, trop long de refaire tous les calculs).

Avec les paramètres par défaut des deux algorithmes on peut constater de meilleurs résultats avec le NL Bayes par rapport au BM3D sur l'ensemble des Sigma, et l'écart est encore plus grand sur les bruits de faible intensité.

Cela s'explique en partie à cause du pas de décalage = 1 dans le NL Bayes alors qu'il est de 3 dans BM3D (ça accélère les calculs par 9, en effet NL Bayes est très long à faire tourner, BM3D un peu moins long de ce fait).

Pour cette raison on recommence l'expérience (sur l'image Barbara.png) avec BM3D en fixant le pas à 1 pour voir si cela change la donne.

BM3D - Barbara

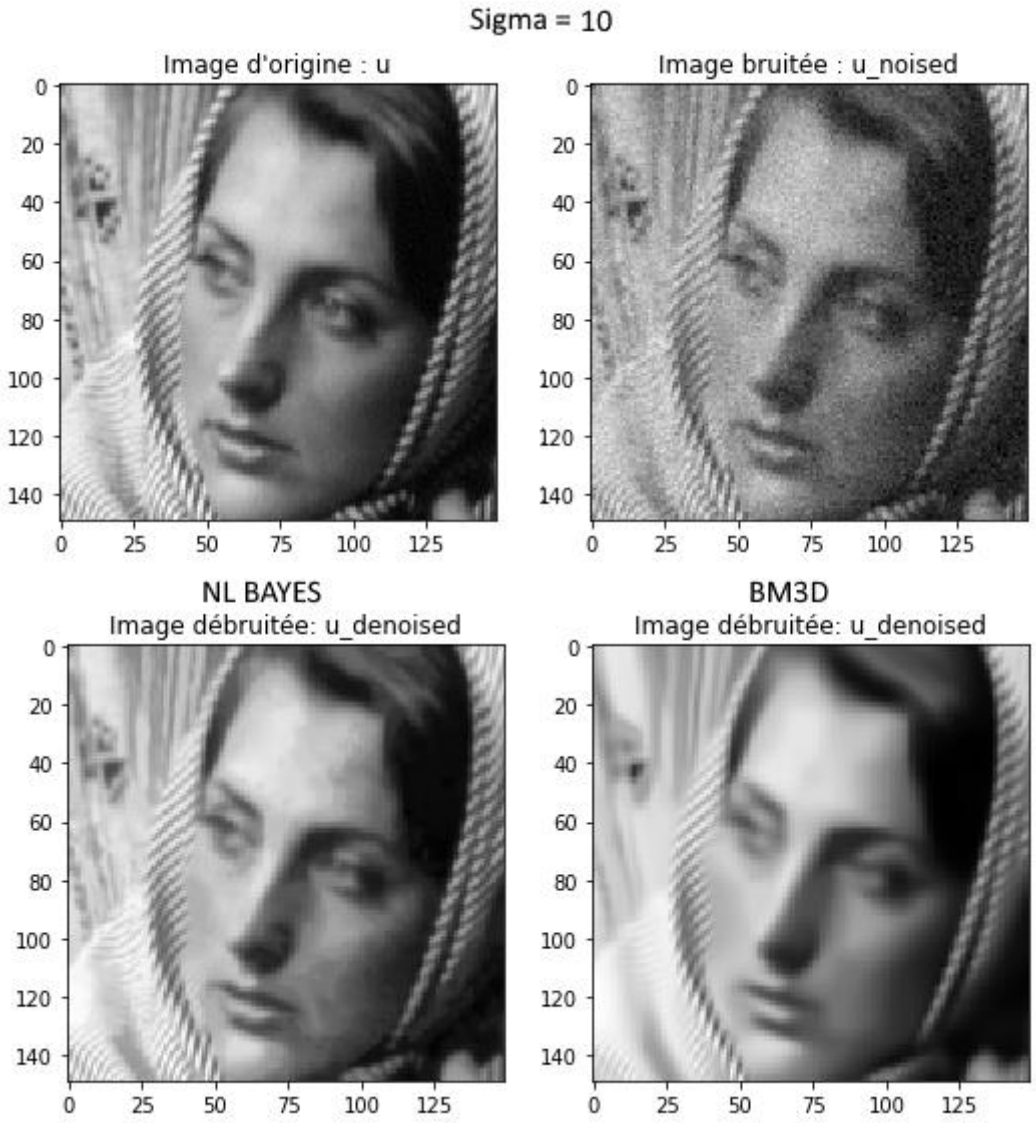
Sigma	PSNR noised	PSNR denoised
2	43.8	30.9
5	34.8	30.5
10	28.4	29.6
20	22.3	27.9
30	18.9	26.8
40	16.5	25.9
60	13.5	24.3
80	11.5	23.0
100	10.2	21.9

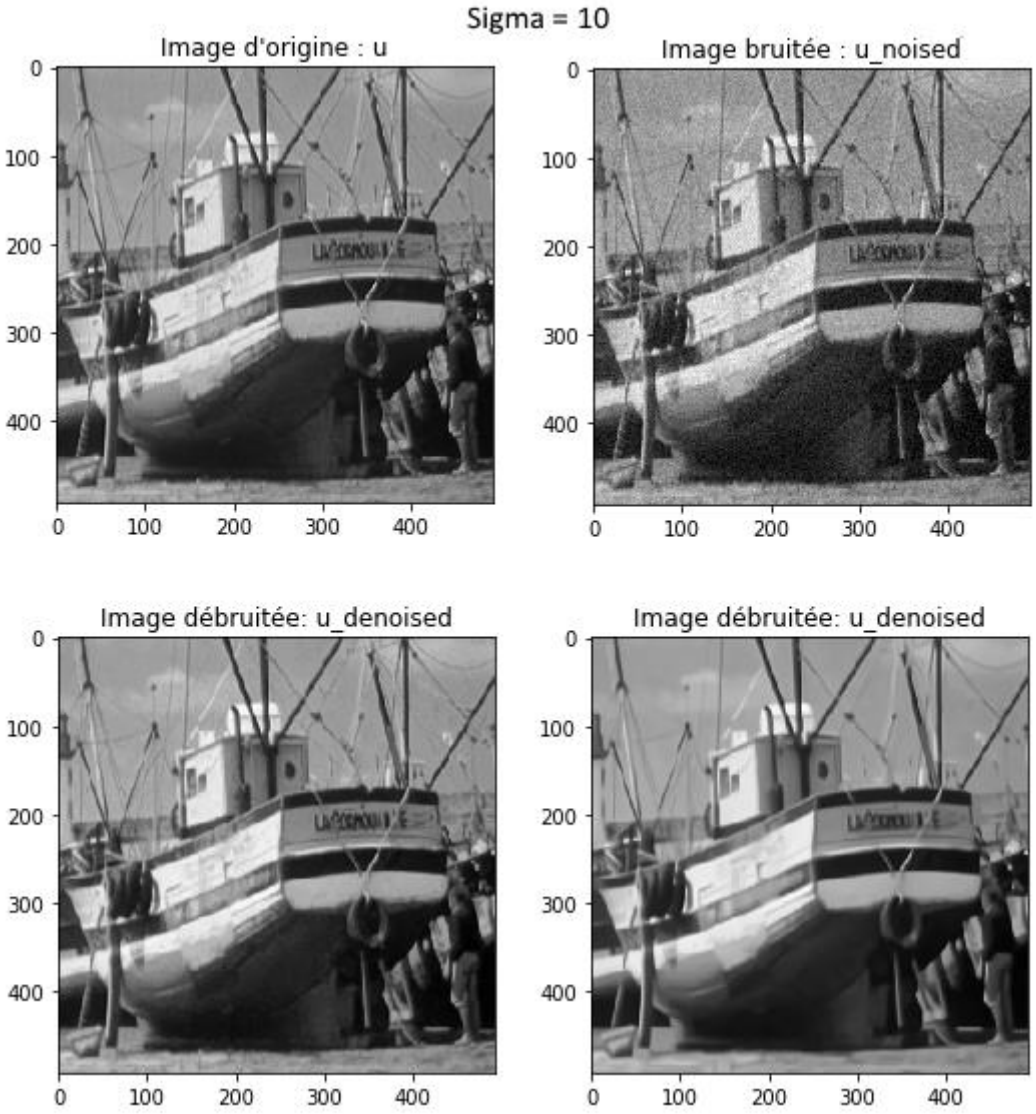
Même avec un pas = 1 l'algorithme BM3D restitue des photos de légèrement moins bonne qualité le NL Bayes.

Comparaison des algorithmes :

Comme on a pu le constater précédemment à l'aide du PSNR et du RMSE, NL Bayes restitue (avec un bruit gaussien) des photos d'une qualité un peu supérieure à celles restitués par le BM3D.

Visuellement cela se constate aussi, le NL Bayes restitue des images qui semblent mieux conserver les lumières et les ombres et paraissent un peu moins flou.





Voir plus en annexe et dans les 2 notebook dans la partie "experiment"

Comparaison des algorithmes avec des techniques plus simple

On constate pour qu'avec un bruit gaussien (où sigma = 10) BM3D et NL Bayes restituent les images avec une meilleure qualité que les filtres classiques.

Sigma = 10	PSNR noised	PSNR denoised
Median filter	28.5	26
Maximum filter	28.5	22.4
Gaussian filter	28.5	20.2
BM3D	28.5	29.5
NL Bayes	28.5	33.0

Voir image filtre en annexe ou notebook partie experiment si besoin

Rapide conclusion sur les résultats expérimentaux

Au travers des expériences qu'on a mené on a pu constater avec les paramètres par défaut de chacun des algorithmes de meilleurs résultats mais légers, pour les bruits de faible intensité. L'algorithme NL Bayes semble également meilleur bien que BM3D ait déjà de très bons résultats par rapport à d'autres algorithmes de débruitage.

Conclusion

Nous avons donc au cours de ce projet implémenter deux algorithmes de débruitage très similaires dans leur structure : l'algorithme BM3D et l'algorithme NL-Bayes. Les deux algorithmes se différencient sur la phase de collaborating filtering justifiant leur fondement théorique. En effet, NL-Bayes se base sur le théorème de Bayes et BM3D sur un filtre de Wiener et la transformation 3D. Les

Suite aux expériences menées et les résultats (non exhaustifs) obtenus, il semble que l'algorithme NL-Bayes ait un léger avantage en termes de qualité de débruitage.

Références :

BM3D: https://www.ipol.im/pub/art/2012/l-bm3d/article.pdf

NL-BAYES: https://www.ipol.im/pub/art/2013/16/article.pdf

Secrets of image denoising cuisine:

http://mcolom.perso.math.cnrs.fr/download/articles/acta_numerica.pdf?fbclid=IwAR23wScDWCFEZhrtUWb7j1tI6xjOyUpUoEa8_EEbM2mFjxiKhGljBGHs_4o