



# Projet Image

## Débruitage d'images

### Compte-rendu n°4

Adam Bonbon

Alaric Hunot-Martin

Louis Jean

Master 2 IMAGINE  
Université de Montpellier

10 novembre 2024

## Table des matières

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Mesure de performance de certains algorithmes de débruitage classiques et implémentation de nouvelles méthodes</b>	<b>2</b>
2.1	Filtre moyenneur . . . . .	2
2.2	Méthode de Fourier . . . . .	3
2.3	Méthode basée sur la densité de probabilité . . . . .	5
2.4	Méthode de Wiener . . . . .	6
2.5	Méthode Non-Local Means (NLM) . . . . .	7
2.6	Méthode BM3D . . . . .	8
2.7	Méthode Guidé . . . . .	9
2.8	Méthode Bilatéral Non Adaptatif . . . . .	10
2.9	Filtrage Bilatéral Adaptatif . . . . .	11
<b>3</b>	<b>Première exploration de CGNet</b>	<b>12</b>
3.1	Tentative d'utiliser le modèle déjà pré-entraîné . . . . .	12
3.2	Réimplémentation basique du modèle . . . . .	13
<b>4</b>	<b>Bases de données</b>	<b>15</b>
<b>5</b>	<b>Réécriture des méthodes classiques et premières comparaisons</b>	<b>15</b>
<b>6</b>	<b>Début d'implémentation de l'interface graphique</b>	<b>18</b>
<b>7</b>	<b>Conclusion</b>	<b>19</b>
<b>8</b>	<b>Perspectives</b>	<b>19</b>
<b>9</b>	<b>Références</b>	<b>19</b>

# 1 Introduction

Cette semaine, nous avons exploré CGNet et amélioré nos mesures de performances pour les méthodes classiques. Aussi, nous avons commencé à designer une application.

## 2 Mesure de performance de certains algorithmes de débruitage classiques et implémentation de nouvelles méthodes

Nous avons évalué les performances de trois algorithmes de débruitage que nous avons déjà implémenté : le filtre moyenneur, la méthode avec la transformée de Fourier et la méthode basée sur la densité de probabilité. Pour chaque méthode, nous avons bruité 16 images en utilisant deux types de bruit : gaussier et sel et poivre. Nous avons ensuite calculé le PSNR pour chacune des images débruitée et originale puis avons fait une moyenne. Enfin, nous avons tracé la courbe du PSNR en fonction des paramètres pour analyser et interpréter nos résultats.

### 2.1 Filtre moyenneur

Pour cette méthode, nous avons évalué l'impact de la taille du noyau sur le PSNR pour les deux types de bruit.

#### 2.1.1 Bruit Gaussien

Avec le bruit gaussien, nous avons observé une augmentation progressive du PSNR. La courbe montre une croissance plutôt rapide, jusqu'à un certain point, après quoi l'augmentation devient plus lente. Cela indique qu'au-delà d'une certaine taille de noyau, l'amélioration de la qualité de l'image devient moins significative.

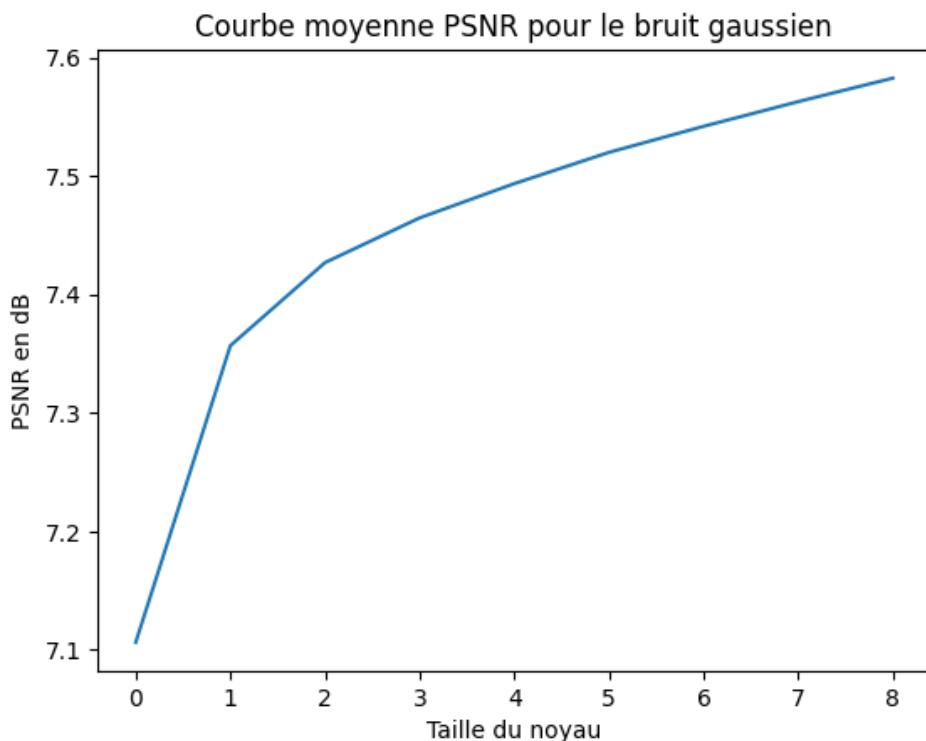


Figure 1: Courbes du PSNR en fonction de la taille du noyau pour le filtre moyenneur avec bruit gaussien

### 2.1.2 Bruit sel et poivre

Pour le bruit sel et poivre, le comportement est similaire à celui observé avec le bruit gaussien, soit une augmentation rapide suivie d'une stabilisation.

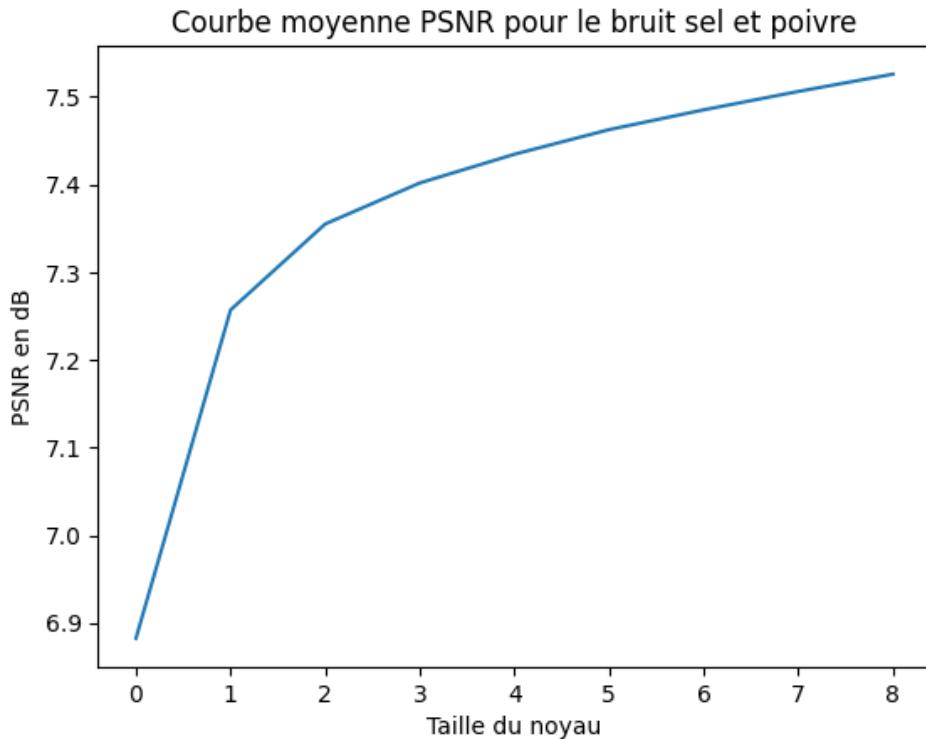


Figure 2: Courbes du PSNR en fonction de la taille du noyau pour le filtre moyenneur avec bruit sel et poivre

## 2.2 Méthode de Fourier

Pour cette méthode, nous avons utilisé la transformée de Fourier pour isoler les basses fréquences à l'aide d'un cercle. Le rayon de ce cercle est le paramètre que nous avons modifié pour observer son effet sur le PSNR

### 2.2.1 Bruit gaussien

Pour le bruit gaussien, on remarque une diminution plus le rayon du cercle augmente. Cela s'explique par le fait que plus le rayon est grand, plus nous incluons les hautes fréquences dans l'image, ce qui rapproche l'image débruitée de l'image bruitée initiale.

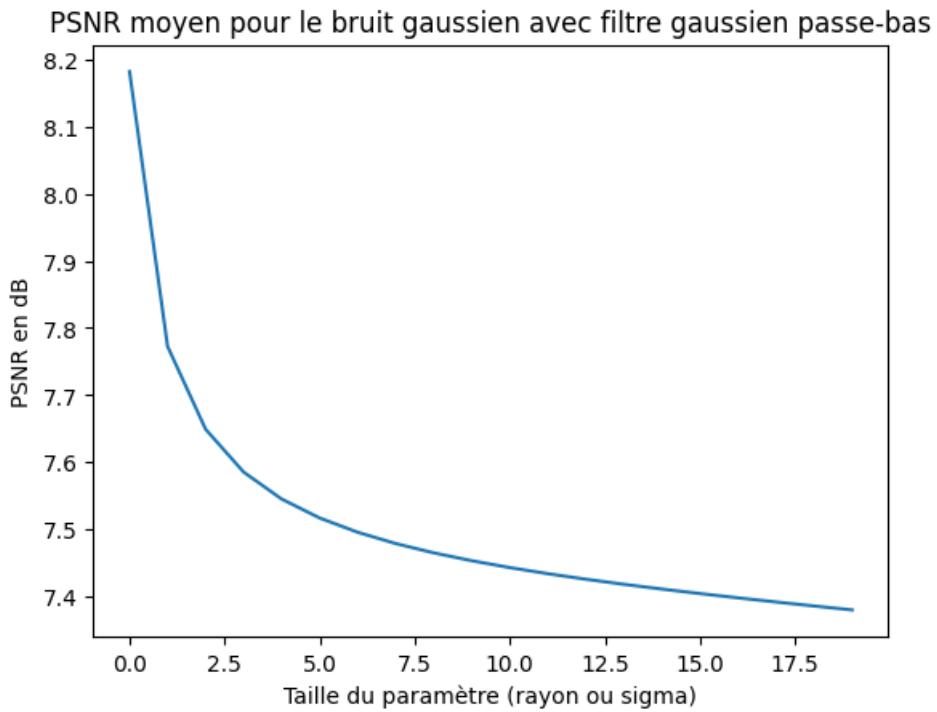


Figure 3: Courbes du PSNR en fonction du rayon du cercle pour la méthode de Fourier avec bruit gaussien

### 2.2.2 Bruit sel et poivre

Pour le bruit sel et poivre, on observe le même comportement et quasiment le résultat. On peut donc en déduire la même interprétation.

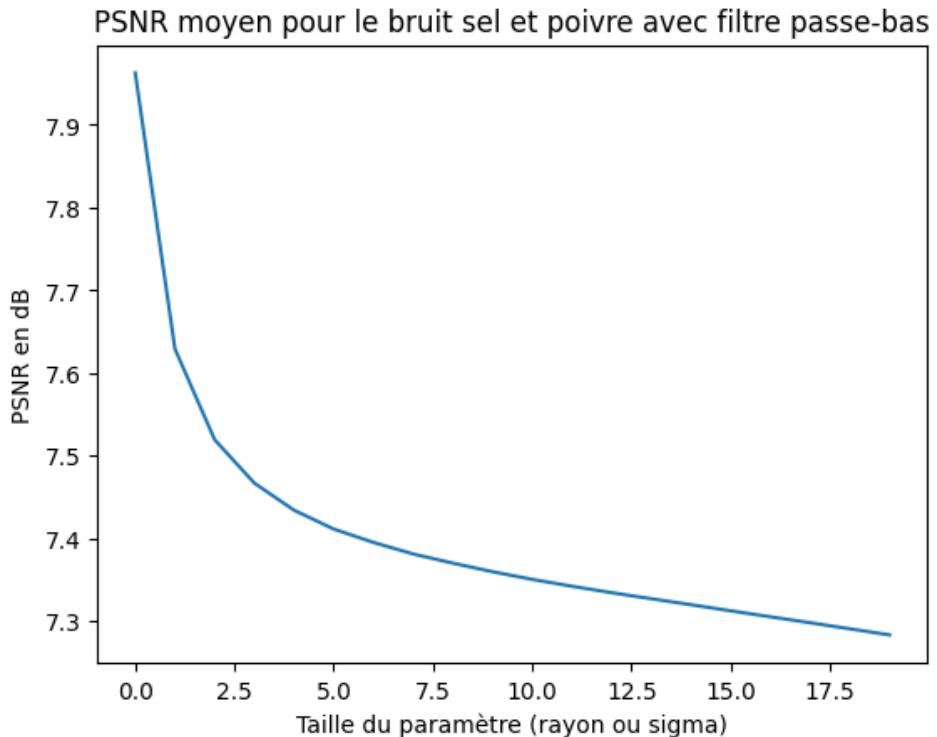


Figure 4: Courbes du PSNR en fonction du rayon du cercle pour la méthode de Fourier avec bruit sel et poivre

## 2.3 Méthode basée sur la densité de probabilité

Dans cette méthode, nous avons observé l'impact du paramètre sigma sur la qualité du débruitage. Le sigma représente l'écart-type de la distribution gaussienne utilisée pour modéliser le bruit.

### 2.3.1 Bruit gaussien

Le PSNR ne s'améliore que marginalement. Après un certain seuil de sigma, la courbe devient presque plate, ce qui signifie que l'algorithme atteint un point où l'augmentation de sigma ne conduit plus à une amélioration significative du PSNR.

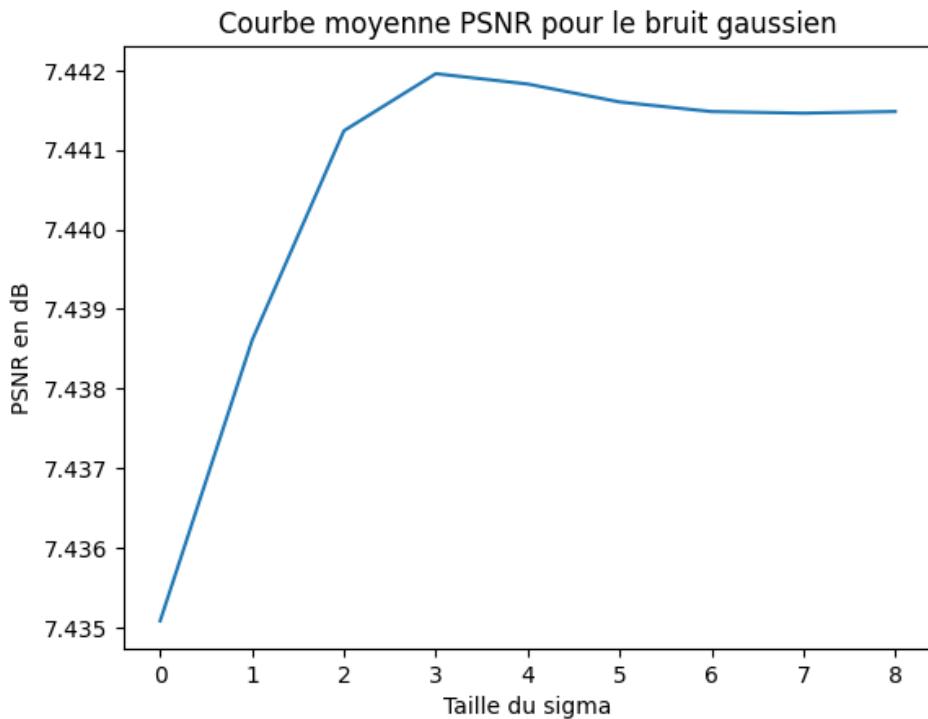


Figure 5: Courbes du PSNR en fonction du paramètre sigma pour la méthode basée sur la densité de probabilité avec bruit gaussien

### 2.3.2 Bruit sel et poivre

Pour le bruit sel et poivre, nous avons constaté une légère augmentation du PSNR, mais cette augmentation est très lente. Cette tendance montre que l'algorithme commence à avoir un impact mais l'effet reste assez limité.

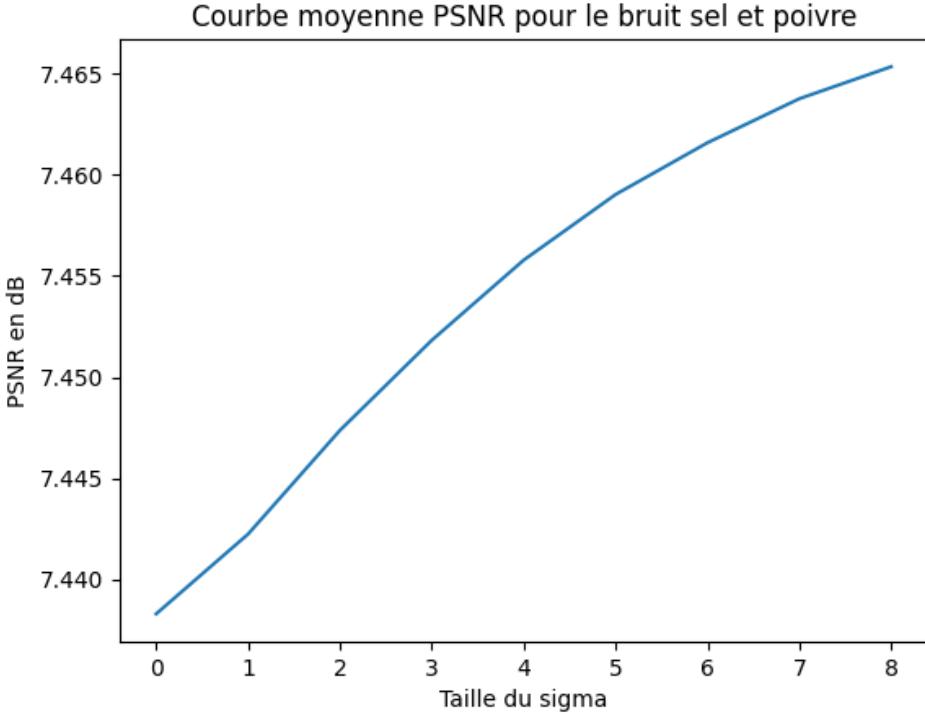


Figure 6: Courbes du PSNR en fonction du paramètre sigma pour la méthode basée sur la densité de probabilité avec bruit sel et poivre

## 2.4 Méthode de Wiener

### 2.4.1 Filtrage de Wiener

Nous avons également implémenté le filtrage fréquentiel de Wiener pour améliorer le débruitage d'images dans le cadre de notre projet. Le filtre de Wiener est une méthode classique qui vise à réduire le bruit tout en préservant les détails importants de l'image. Il fonctionne en utilisant une approche basée sur la fréquence pour estimer le signal original à partir de l'image bruitée, en minimisant l'erreur quadratique moyenne.

L'implémentation se base sur la *transformée de Fourier* pour analyser l'image dans le domaine fréquentiel. Le filtre de Wiener utilise la formule suivante pour estimer le spectre de l'image débruitée :

$$H(u, v) = \frac{S_{xx}(u, v)}{S_{xx}(u, v) + S_{nn}(u, v)}$$

où :

- $H(u, v)$  est le filtre de Wiener.
- $S_{xx}(u, v)$  représente la puissance spectrale du signal d'origine.
- $S_{nn}(u, v)$  est la puissance spectrale du bruit.

Le filtre est appliqué dans le domaine fréquentiel de la manière suivante :

$$G_{\text{débruité}}(u, v) = H(u, v) \cdot G_{\text{bruité}}(u, v)$$

où :

- $G_{\text{débruité}}(u, v)$  est le spectre de l'image après application du filtre.
- $G_{\text{bruité}}(u, v)$  est le spectre de l'image bruitée.

Cette méthode est particulièrement efficace pour le débruitage d'images contaminées par du bruit gaussien, car elle optimise le rapport signal sur bruit. Le filtre de Wiener ajuste dynamiquement son action en fonction de la fréquence spatiale, permettant ainsi de conserver les hautes fréquences qui correspondent aux détails importants de l'image, tout en réduisant les basses fréquences associées au bruit.

#### 2.4.2 Bruit gaussien

L'application de ce filtre nous a permis d'obtenir des résultats encourageants, avec une amélioration notable du *PSNR* par rapport aux images bruitées. La courbe PSNR obtenue montre une augmentation moyenne de la qualité d'image après filtrage, bien que les performances varient selon le niveau de bruit initial et le contenu de chaque image.

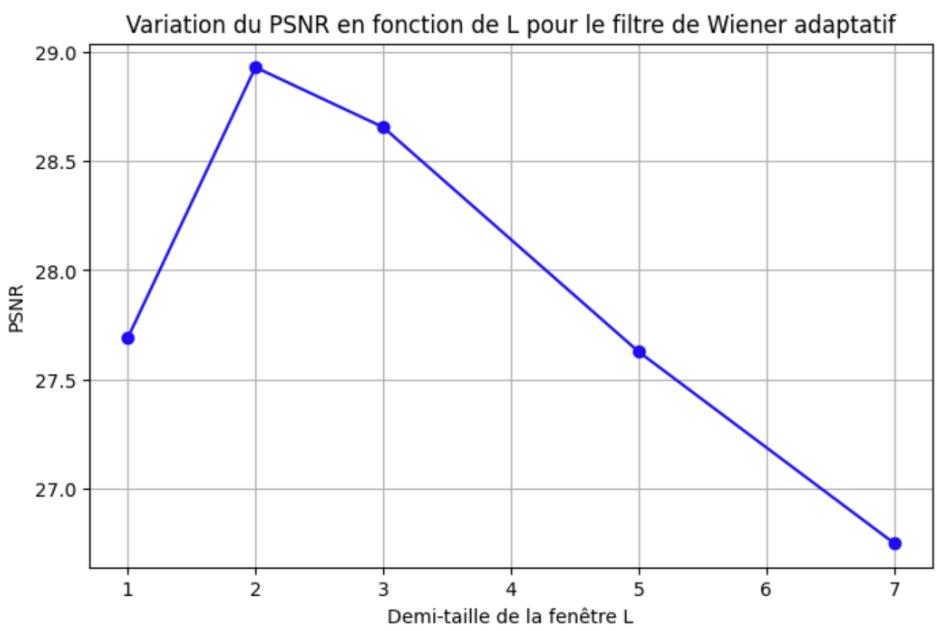


Figure 7: Courbe PSNR obtenue après application du filtre de Wiener

## 2.5 Méthode Non-Local Means (NLM)

### 2.5.1 Filtrage NLM

Nous avons également implémenté le filtrage Non-Local Means (NLM) pour améliorer le débruitage des images. Cette méthode est basée sur l'idée que les pixels similaires d'une image, même s'ils sont éloignés les uns des autres, peuvent être utilisés pour estimer un pixel donné. Contrairement aux filtres classiques qui ne tiennent compte que des pixels voisins immédiats, le filtre NLM cherche à exploiter la redondance des motifs dans toute l'image.

Le filtrage NLM utilise la formule suivante pour estimer la valeur d'un pixel débruité  $I_{\text{débruité}}(i, j)$  :

$$I_{\text{débruité}}(i, j) = \frac{\sum_{k,l} w((i, j), (k, l)) \cdot I(k, l)}{\sum_{k,l} w((i, j), (k, l))}$$

où :

- $I(k, l)$  est l'intensité du pixel à la position  $(k, l)$ .
- $w((i, j), (k, l))$  est le poids qui détermine la similarité entre les pixels  $(i, j)$  et  $(k, l)$ .

Le poids  $w((i, j), (k, l))$  est calculé en utilisant une fonction gaussienne basée sur la distance euclidienne entre les patchs autour des pixels  $(i, j)$  et  $(k, l)$  :

$$w((i,j), (k,l)) = \exp\left(-\frac{\|P_{i,j} - P_{k,l}\|^2}{h^2}\right)$$

où :

- $P_{i,j}$  et  $P_{k,l}$  sont les patchs centrés respectivement sur les pixels  $(i,j)$  et  $(k,l)$ .
- $\|P_{i,j} - P_{k,l}\|^2$  est la distance euclidienne entre les deux patchs.
- $h$  est un paramètre de lissage qui contrôle la sensibilité du filtre.

### 2.5.2 Bruit gaussien

L'application du filtre NLM a montré des résultats prometteurs, comme illustré par la courbe ci-dessous. En ajustant la valeur du paramètre  $h$ , nous avons observé que le *PSNR* atteint un maximum autour de  $h = 0.10$  avant de diminuer lorsque  $h$  augmente.

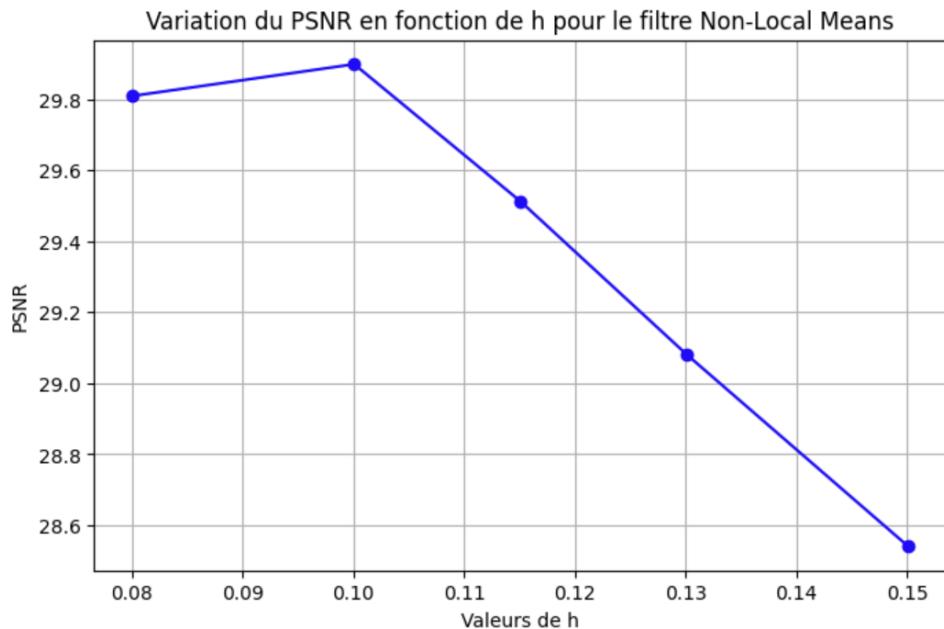


Figure 8: Variation du PSNR en fonction du paramètre  $h$  pour le filtre Non-Local Means

## 2.6 Méthode BM3D

### 2.6.1 Filtre BM3D

La méthode BM3D fonctionne en deux étapes principales :

1. \*\*Regroupement et filtrage collaboratif\*\* : Les blocs similaires sont identifiés à travers l'image et regroupés en un volume 3D. Un filtrage dans le domaine fréquentiel est appliqué à ce volume pour réduire le bruit.
2. \*\*Agrégation et reconstruction\*\* : Les blocs filtrés sont replacés à leurs positions d'origine, puis une agrégation des résultats est effectuée pour obtenir l'image finale.

La formule utilisée pour appliquer le filtre BM3D peut être résumée comme suit :

$$\hat{I} = \sum_i w_i \cdot \mathcal{F}^{-1}(H_i \cdot \mathcal{F}(B_i))$$

où :

- $\hat{I}$  est l'image débruitée.
- $B_i$  représente les blocs extraits de l'image.
- $\mathcal{F}$  et  $\mathcal{F}^{-1}$  sont respectivement la transformée de Fourier et sa transformée inverse.
- $H_i$  est le filtre appliqué dans le domaine fréquentiel.
- $w_i$  est un poids qui dépend de la similarité des blocs.

### 2.6.2 Bruit gaussien

Nous avons analysé l'impact du paramètre  $\sigma_{\text{psd}}$ , qui contrôle le niveau de bruit supposé, sur le *PSNR*. La courbe ci-dessous montre que le *PSNR* diminue à mesure que  $\sigma_{\text{psd}}$  augmente, ce qui signifie que le débruitage devient moins efficace lorsque le niveau de bruit est élevé. Le paramètre optimal se situe autour de valeurs faibles de  $\sigma_{\text{psd}}$ , où le PSNR atteint un maximum d'environ 32 dB.

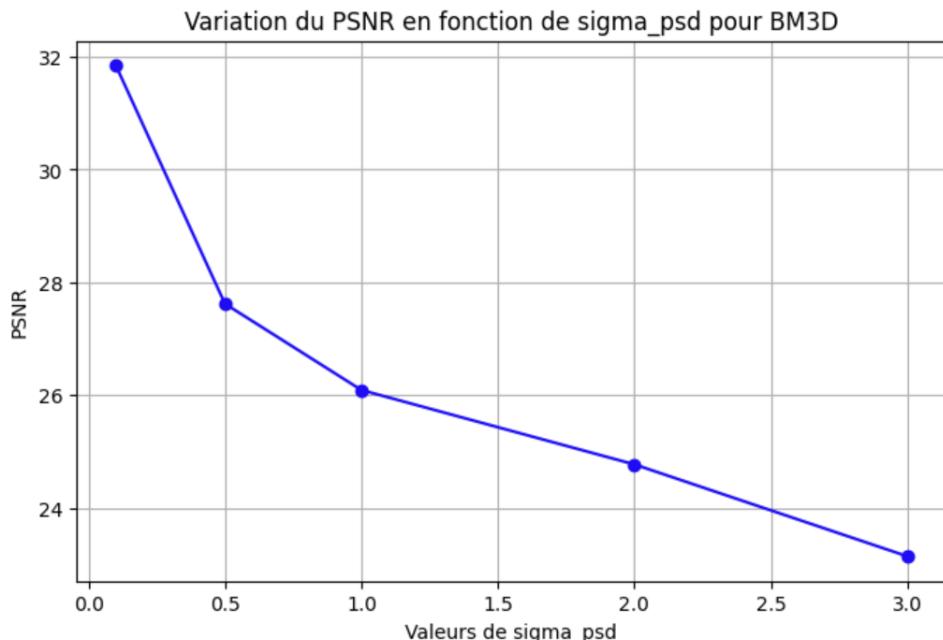


Figure 9: Variation du PSNR en fonction de  $\sigma_{\text{psd}}$  pour le filtre BM3D

## 2.7 Méthode Guidé

### 2.7.1 Filtrage guidé

Le filtrage guidé (Guided Filter) est une technique de filtrage d'images qui utilise une image de guidance pour améliorer la qualité du filtrage. Contrairement aux filtres traditionnels, le filtre guidé est capable de préserver les contours et les détails tout en lissant les régions homogènes. Cela en fait un choix efficace pour des applications comme le débruitage d'images, l'amélioration de la netteté et la suppression des artefacts.

Le filtrage guidé utilise la formule suivante pour estimer la valeur d'un pixel filtré  $q(i, j)$  :

$$q(i, j) = a(i, j)I(i, j) + b(i, j)$$

où :

- $I(i, j)$  est l'intensité de l'image de guidance.
- $a(i, j)$  et  $b(i, j)$  sont des coefficients locaux calculés pour minimiser la différence entre l'image d'entrée et l'image filtrée.

Les coefficients  $a(i, j)$  et  $b(i, j)$  sont définis par :

$$a(i, j) = \frac{\text{Cov}(I, p)}{\sigma_I^2 + \epsilon}, \quad b(i, j) = \bar{p} - a(i, j)\bar{I}$$

où :

- $\text{Cov}(I, p)$  est la covariance entre l'image de guidance  $I$  et l'image d'entrée  $p$ .
- $\sigma_I^2$  est la variance de l'image de guidance dans une fenêtre locale.
- $\epsilon$  est un paramètre de régularisation pour contrôler le lissage.
- $\bar{I}$  et  $\bar{p}$  sont les moyennes locales de  $I$  et  $p$ .

### 2.7.2 Bruit gaussien

En utilisant le filtre guidé, nous avons analysé l'impact du paramètre  $\epsilon$  sur la qualité du débruitage. La courbe ci-dessous montre que le *PSNR* (Peak Signal-to-Noise Ratio) atteint un maximum pour des valeurs faibles de  $\epsilon$ , puis diminue à mesure que  $\epsilon$  augmente, ce qui indique que le filtre lisse excessivement les détails fins lorsque  $\epsilon$  est trop élevé.

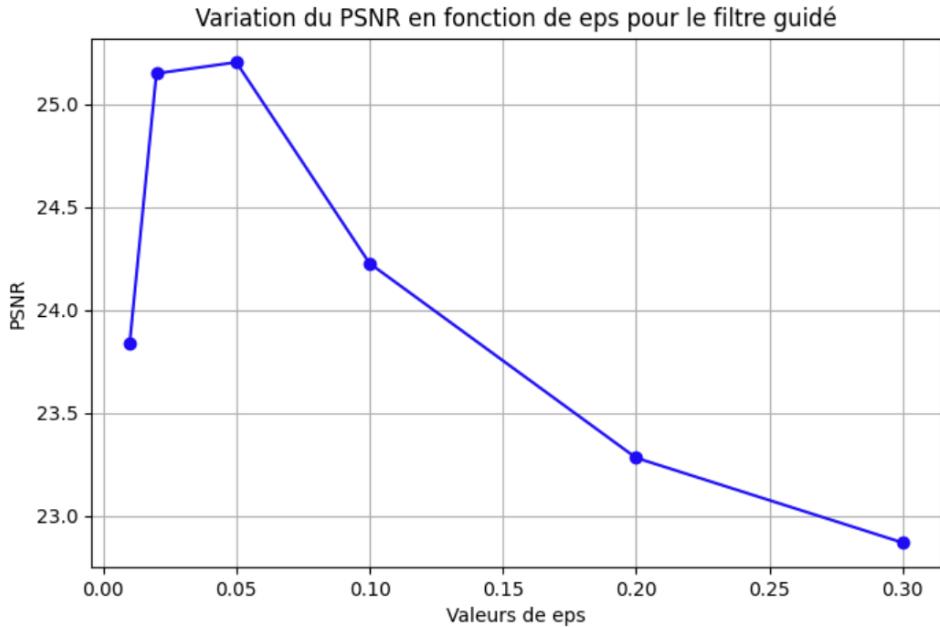


Figure 10: Variation du PSNR en fonction de  $\epsilon$  pour le filtre guidé

## 2.8 Méthode Bilatéral Non Adaptatif

### 2.8.1 Filtrage bilatéral non adaptatif

Le filtre bilatéral est une technique de débruitage non linéaire qui préserve les contours en prenant en compte à la fois la proximité spatiale des pixels et leur similarité d'intensité. Contrairement aux filtres gaussiens classiques qui lissent uniformément, le filtre bilatéral permet de réduire le bruit tout en conservant les bords importants dans l'image.

La formule du filtre bilatéral pour un pixel  $i$  est donnée par :

$$I_{\text{filtré}}(i) = \frac{1}{W_i} \sum_{j \in \Omega} I(j) \cdot \exp \left( -\frac{\|x_i - x_j\|^2}{2\sigma_s^2} \right) \cdot \exp \left( -\frac{|I(i) - I(j)|^2}{2\sigma_{\text{color}}^2} \right)$$

où :

- $I_{\text{filtré}}(i)$  est l'intensité du pixel filtré.
- $I(j)$  est l'intensité du pixel voisin  $j$ .
- $W_i$  est un facteur de normalisation pour assurer que la somme des poids soit égale à 1.
- $\Omega$  est la fenêtre locale autour du pixel  $i$ .
- $\sigma_s$  contrôle l'influence de la distance spatiale.
- $\sigma_{\text{color}}$  contrôle l'influence de la différence d'intensité entre les pixels.

Le paramètre clé  $\sigma_{\text{color}}$  joue un rôle essentiel dans le filtrage, comme illustré par la courbe ci-dessous.

### 2.8.2 Bruit gaussien

En utilisant différentes valeurs pour  $\sigma_{\text{color}}$ , nous avons observé une amélioration du *PSNR* (Peak Signal-to-Noise Ratio) jusqu'à un certain point, après lequel le *PSNR* commence à diminuer. Cela indique que le choix du paramètre  $\sigma_{\text{color}}$  est crucial pour atteindre un bon compromis entre le lissage du bruit et la préservation des détails.

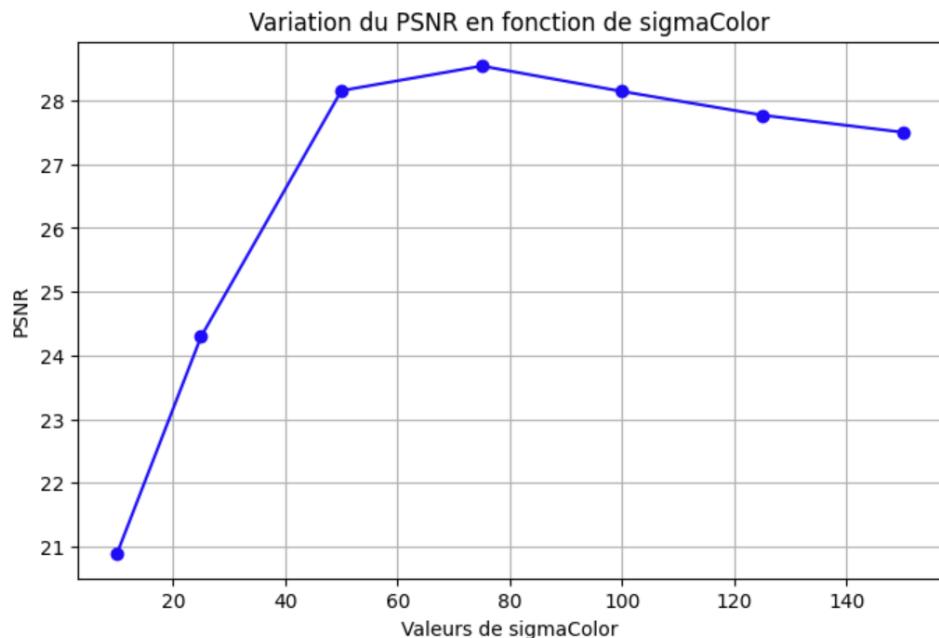


Figure 11: Variation du PSNR en fonction de  $\sigma_{\text{color}}$  pour le filtre bilatéral

## 2.9 Filtrage Bilatéral Adaptatif

### 2.9.1 Formule et Différence avec le Filtrage Non Adaptatif

Le filtrage bilatéral adaptatif est une extension du filtre bilatéral classique. Contrairement au filtrage non adaptatif où les paramètres  $\sigma_s$  et  $\sigma_{\text{color}}$  sont constants pour toute l'image, le filtre bilatéral adaptatif ajuste dynamiquement ces paramètres en fonction des caractéristiques locales de chaque pixel.

La formule du filtre bilatéral adaptatif pour un pixel  $i$  est donnée par :

$$I_{\text{filtré}}(i) = \frac{1}{W_i} \sum_{j \in \Omega} I(j) \cdot \exp \left( -\frac{\|x_i - x_j\|^2}{2\sigma_s(i)^2} \right) \cdot \exp \left( -\frac{|I(i) - I(j)|^2}{2\sigma_{\text{color}}(i, j)^2} \right)$$

où :

- $I_{\text{filtré}}(i)$  est l'intensité du pixel filtré.
- $I(j)$  est l'intensité du pixel voisin  $j$ .
- $W_i$  est un facteur de normalisation pour assurer que la somme des poids soit égale à 1.
- $\Omega$  est la fenêtre locale autour du pixel  $i$ .
- $\sigma_s(i)$  est le paramètre spatial qui varie en fonction de la position du pixel  $i$ .
- $\sigma_{\text{color}}(i, j)$  est un paramètre adaptatif basé sur la différence d'intensité entre les pixels  $i$  et  $j$ .

L'avantage principal du filtrage bilatéral adaptatif est qu'il permet de préserver les détails dans les zones à fort contraste tout en supprimant le bruit dans les régions homogènes. Le réglage dynamique des paramètres en fonction de la variance locale améliore la performance par rapport au filtre bilatéral standard.

### 2.9.2 Problèmes avec la Méthode

Bien que le filtrage bilatéral adaptatif offre de meilleures performances pour la préservation des contours, il présente plusieurs inconvénients :

- Cette méthode est particulièrement adaptée pour des images de \*\*petite taille\*\* car elle est \*\*très coûteuse en temps de calcul\*\*. En effet, l'ajustement dynamique des paramètres pour chaque pixel nécessite des calculs intensifs.
- Dans notre cas, il a fallu \*\*28 minutes\*\* pour obtenir une seule image PGM de taille  $512 \times 512$  en utilisant cette méthode, ce qui n'est pas pratique pour des applications en temps réel ou pour des images de plus grande taille.
- La complexité de cette méthode la rend difficile à optimiser pour des systèmes avec des ressources limitées, comme des appareils mobiles ou des environnements embarqués.

## 3 Première exploration de CGNet

### 3.1 Tentative d'utiliser le modèle déjà pré-entraîné

La base de code de CGNet a été récupérée depuis le dépôt suivant :

<https://github.com/Ascend-Research/CascadedGaze>.

Le dataset utilisé pour entraîner le modèle est le **Smartphone Image Denoising (SIDD)**, une base de données d'images bruitées provenant de 10 scènes différentes, avec des conditions d'éclairage variées, capturées à l'aide de cinq caméras de smartphone.

L'entraînement du modèle CGNet s'est avéré particulièrement coûteux en ressources. L'une des principales difficultés rencontrées réside dans la gestion de la puissance de calcul, étant donné les ressources limitées disponibles sur notre machine personnelle. L'entraînement était possible mais il aurait fallu accepter des délais très longs, dépassant une semaine, avant d'obtenir des résultats concrets.

L'objectif serait de rendre l'entraînement du modèle plus accessible en réduisant le temps nécessaire pour obtenir des résultats convaincants.

## 3.2 Réimplémentation basique du modèle

Nous avons tenté de créer un réseau de neurones type CGNet pour le débruitage en utilisant le dataset BSDS500. Bien que notre implémentation ne soit pas encore totalement fidèle à l'architecture CGNet, elle présente des différences par rapport à un simple CNN traditionnel.

### Ce qui rapproche notre réseau d'un CGNet :

**Convolutions en profondeur :** Notre réseau utilise des convolutions à plusieurs niveaux de profondeur, ce qui permet de capturer des informations contextuelles à différentes échelles. Contrairement à un simple CNN qui se contente de convolutions de base, nous intégrons des couches supplémentaires pour étendre la *receptive field* du modèle, s'inspirant du concept de *context aggregation* présent dans CGNet.

**Utilisation de fonctions d'activation avancées :** Nous avons expérimenté avec des fonctions d'activation et de normalisation pour améliorer la capture des caractéristiques. Bien que cela ne soit pas aussi sophistiqué que le *context module* dans CGNet, cela nous rapproche d'une architecture plus robuste que celle d'un CNN classique.

**Blocs de résidus partiels :** Bien que l'implémentation ne soit pas complète, nous avons partiellement intégré des blocs résiduels pour favoriser la propagation des gradients et améliorer la convergence du modèle. Cela diffère d'un CNN standard qui ne possède généralement pas ces connexions résiduelles.

### Ce qui manque pour être un CGNet complet :

**Modules contextuels :** Dans un CGNet complet, les *context modules* jouent un rôle crucial en agrégeant l'information contextuelle à différentes échelles spatiales pour la segmentation ou le débruitage. Actuellement, notre réseau n'inclut pas ces modules.

**Supervision multi-échelle :** Nous n'avons pas encore intégré de supervision à plusieurs niveaux pour guider l'apprentissage du réseau, ce qui est une des forces de CGNet pour mieux capter les détails fins dans les images bruitées.

**Utilisation de convolutions dilatées :** CGNet utilise des convolutions dilatées pour augmenter la *receptive field* sans perte de résolution, ce que notre réseau ne fait pas encore. Cela limite la capacité de notre modèle à capturer des détails globaux tout en préservant la structure fine.

### Performances actuelles :

Pour l'instant, les résultats obtenus sont encourageants, avec une augmentation du *PSNR* allant de 4 à 10 dB, et une moyenne d'environ 8 dB lorsque l'on compare le *PSNR* entre l'image originale et l'image débruitée par rapport à celui entre l'image originale et l'image bruitée. Cela indique que notre modèle parvient déjà à améliorer significativement la qualité des images. Toutefois, il est important de noter que l'entraînement n'a pas encore été totalement optimisé et que nous devons encore ajuster plusieurs hyperparamètres pour obtenir des performances maximales. Une optimisation supplémentaire de l'architecture et des paramètres devrait permettre d'améliorer encore davantage ces résultats.

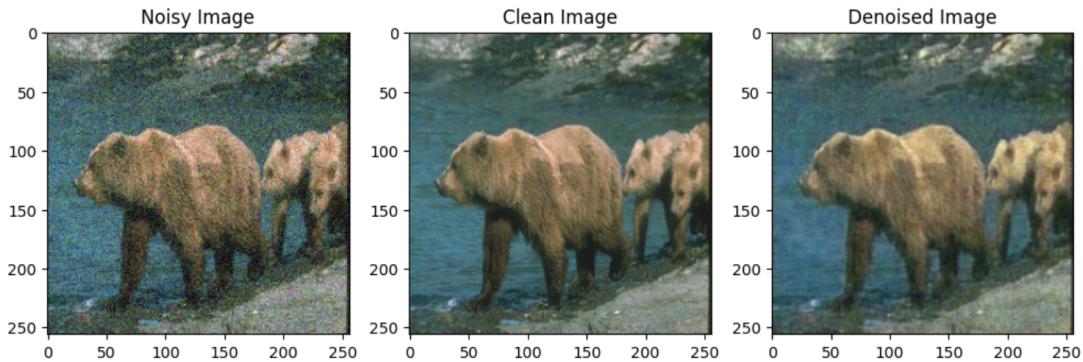


Figure 12: Résultats image 1 de BSDS500  
*PSNR bruitée = 20.10, PSNR débruitée = 28.11*

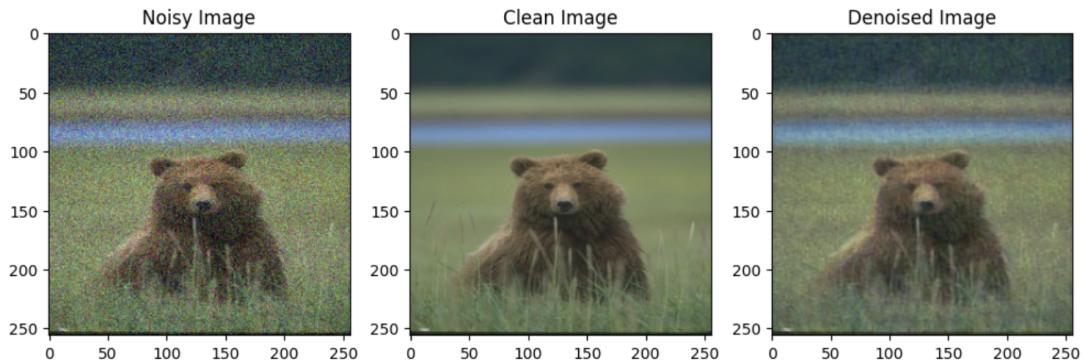


Figure 13: Résultats image 2 de BSDS500  
*PSNR bruitée = 20.06, PSNR débruitée = 29.87*

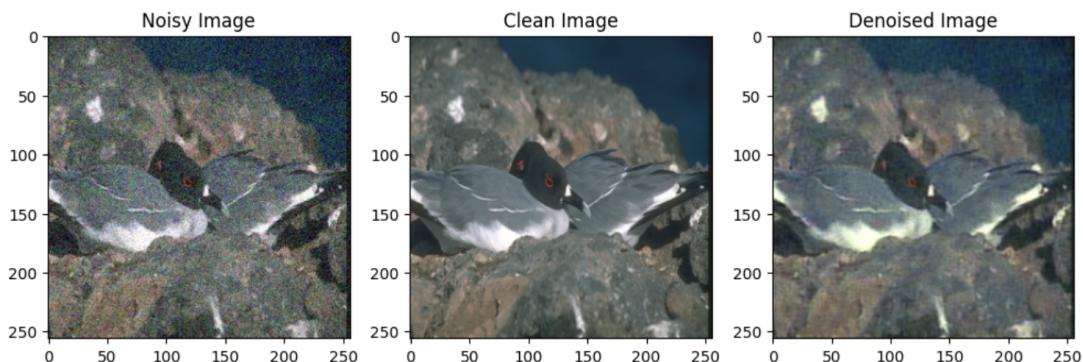


Figure 14: Résultats image 4 de BSDS500  
*PSNR bruitée = 20.12, PSNR débruitée = 28.70*

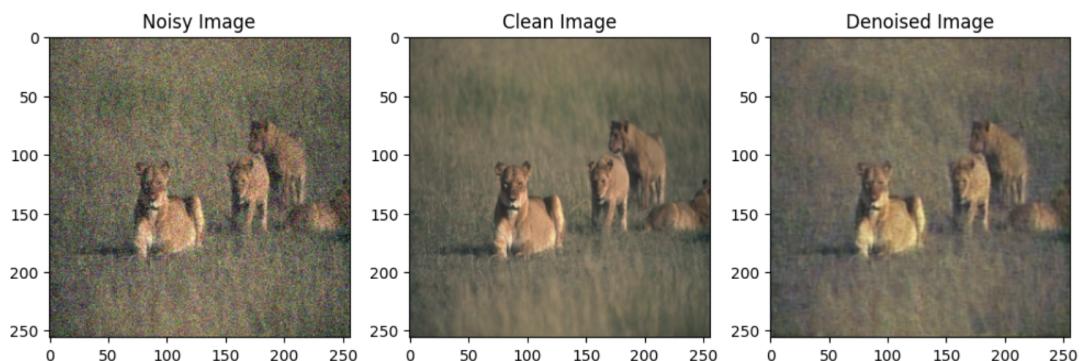


Figure 15: Résultats image 7 de BSDS500  
*PSNR bruitée = 20.01, PSNR débruitée = 30.24*

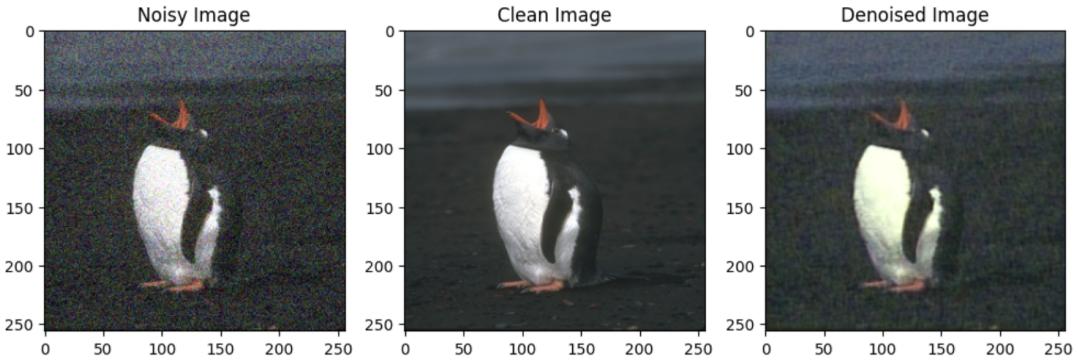


Figure 16: Résultats image 9 de BSDS500  
 $PSNR$  bruitée = 20.25,  $PSNR$  débruitée = 29.46

## 4 Bases de données

Nous avons choisi comme dataset BSDS500, qui est composé d’images d’animaux et d’environnements naturels variés. Nous avons sélectionné ce dataset car, en examinant la littérature scientifique, nous avons constaté qu’il est particulièrement compatible avec une architecture CGNet. L’un des problèmes des autres datasets disponibles est qu’ils sont souvent trop thématiques, ce qui pourrait biaiser les résultats de notre projet. En effet, comme notre objectif est de réaliser du débruitage d’images de manière générale, un dataset thématique risquerait de limiter la diversité des niveaux de gris, aboutissant à des images aux caractéristiques similaires dans certaines régions. Cela ne refléterait pas la diversité nécessaire pour un modèle de débruitage robuste. À l’inverse, BSDS500, avec son contenu varié et ses 200 images, s’est révélé être un choix plus exhaustif et adapté. Les images de ce dataset seront ensuite bruitées de diverses façons afin d’entraîner notre modèle à effectuer du débruitage sur des images aux caractéristiques variées.

## 5 Réécriture des méthodes classiques et premières comparaisons

Afin de commencer à comparer les résultats entre différentes méthodes et la méthode CNN, nous avons utilisé la première réimplémentation que nous avons fait de CGNet. Comme nous avons entraîné notre implémentation sur le dataset BSD500 (que nous avons bruité), qui contient des images couleur, nous avons dû adapter les méthodes classiques pour qu’elles puissent être efficaces sur des images couleur. Pour l’instant, nous avons seulement réécrit 3 méthodes : le filtre médian, la transformation en ondelettes de Haar et BM3D. À partir de là, nous avons écrit un programme qui bruite une image, puis qui applique le débruitage selon les 3 méthodes et le CNN sur cette image bruitée, avant de renvoyer le PSNR pour chaque méthode en sortie. Les résultats sont déjà assez convaincants. Quoiqu’il en soit, nous avons déjà pu sortir quelques résultats, qui seront nettement améliorés au fur et à mesure du projet, mais nous avons désormais une bonne base de comparaison des méthodes. Ci-dessous, voici quelques résultats pour l’image *lena.ppm* et l’image *military.jpg*, issue du dataset BSD500.



Figure 17: lena.ppm

Méthode	PSNR (dB)	
	Bruit gaussien	Bruit sel et poivre
CGNet	25,8	25
Filtre médian	25,9	30
Ondelettes de Haar	26	21
BM3D	20,2	18,1

Table 1: Comparaison du PSNR entre l'image originale et débruitée selon les méthodes de débruitage et le type de bruit pour *lena.ppm*



Figure 18: military.jpg

Méthode	PSNR (dB)	
	Bruit gaussien	Bruit sel et poivre
CGNet	28	26,7
Filtre médian	26	31,1
Ondelettes de Haar	25,6	23,4
BM3D	20,1	18,4

Table 2: Comparaison du PSNR entre l'image originale et débruitée selon les méthodes de débruitage et le type de bruit pour *military.jpg*

Globalement, on remarque que CGNet a un vrai potentiel, alors même que l'implémentation que nous en avons fait est simpliste. Le filtre médian est toujours le meilleur lorsqu'il s'agit de bruit sel et poivre. BM3D semble avoir du mal, peut-être que nous devrions revoir notre code pour cette méthode.

## 6 Début d'implémentation de l'interface graphique

Nous avons commencé à mettre en place une interface graphique utilisateur qui servira de support à notre projet. Son but est de permettre une utilisation intuitive et simple des différentes méthodes de débruitage que nous aurons implémenté au cours du projet. L'utilisateur choisira une image déjà bruitée, ou pourra choisir une image neutre puis la brouter avec le bruit de son choix, puis choisira la méthode de débruitage qu'il veut tester. Après le débruitage effectué, nous afficherons quelques métriques d'évaluation comme le PSNR, SSIM, ... Pour l'instant, l'application permet uniquement de choisir une méthode et une image, mais pas encore de brouter une image ou bien de lancer le débruitage.

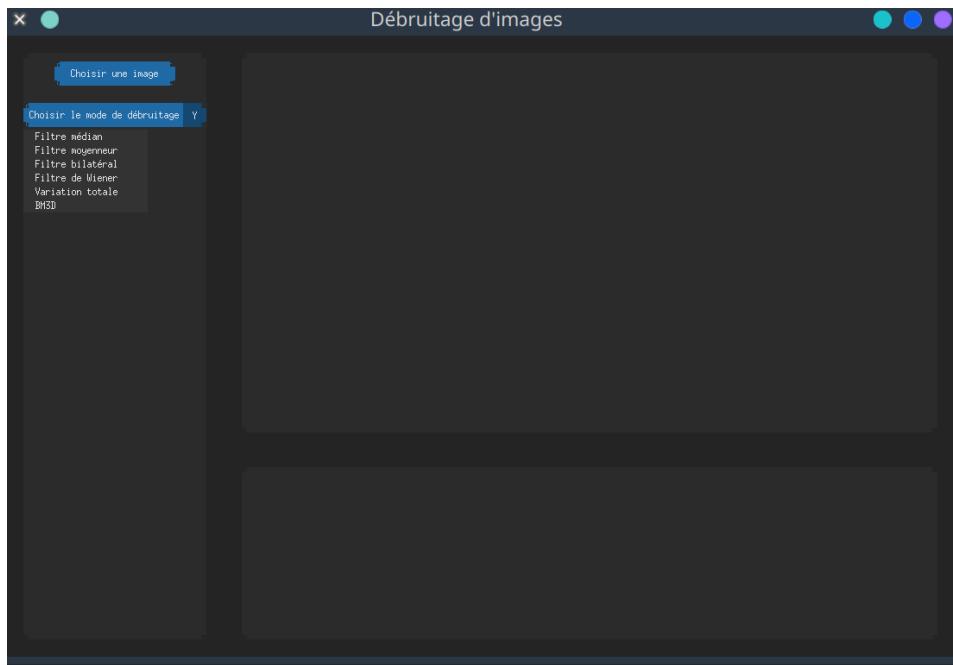


Figure 19: Sélection du mode de débruitage

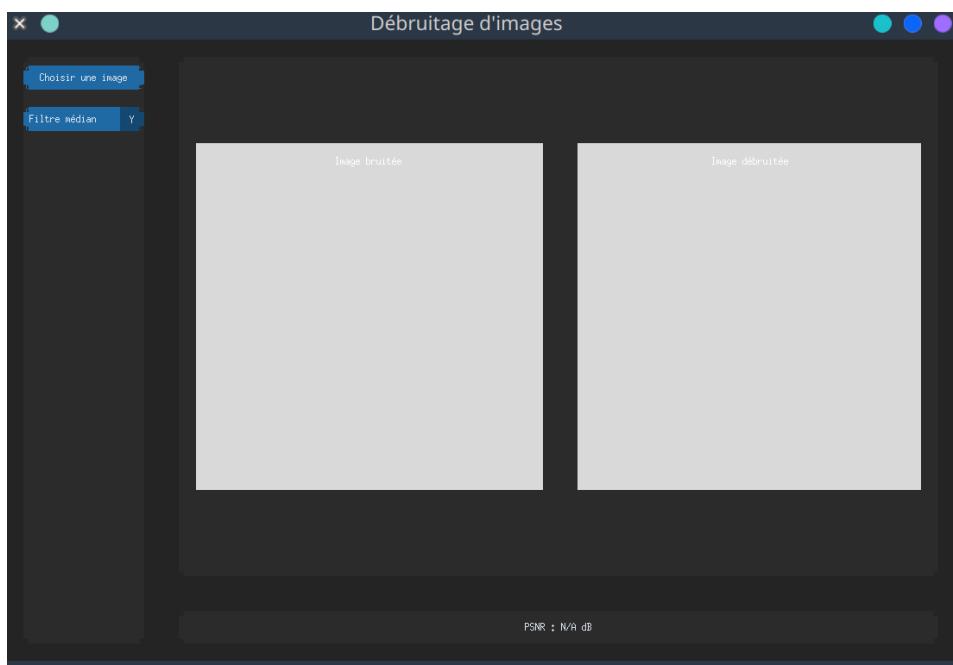


Figure 20: Affichage principal de l'application

## 7 Conclusion

Cette semaine, nous avons commencé à explorer CGNet, et nous avons poussé plus en profondeur les méthodes classiques, notamment en effectuant des mesures de qualité. Nous avons également pu observer les forces et les limites des différentes techniques de débruitage. Ces premières analyses nous ont permis de mieux comprendre les améliorations nécessaires pour nos futures itérations.

## 8 Perspectives

Pour continuer nos travaux actuels, nous allons nous concentrer sur l'optimisation du modèle CGNet, soit en améliorant notre implémentation, soit en renforçant l'entraînement du modèle déjà pré-entraîné. Nous reviendrons certainement sur notre implémentation de BM3D pour voir si l'on peut l'améliorer. Nous continuerons aussi à développer l'interface graphique, en essayant d'avoir au moins une méthode fonctionnelle dans l'application d'ici la semaine prochaine. Nous rajouterons aussi d'autres métriques que le PSNR pour évaluer la qualité de nos débruitages.

Merci pour le temps et l'attention que vous avez consacrés à la lecture de ce compte-rendu.

## 9 Références

Ghasemabadi, A., Janjua, M. K., Salameh, M., Zhou, C., Sun, F., & Niu, D. (2024). *CascadedGaze: Efficiency in Global Context Extraction for Image Restoration*. *Transactions on Machine Learning Research*. <https://openreview.net/forum?id=C3FXHxMVuq>.