Master SDA

Traitement de donnés multimédia

TP 2 : Débruitage des images en python

Encadré par :

- **Pr. Alioua Nawal**

Réalisée par :

- **AGUERCHI Saida**
- **BENAGUERRI Safaa**

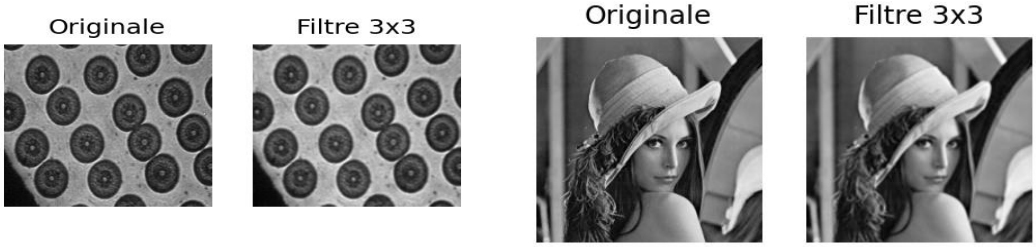
1 Filtre moyenneur

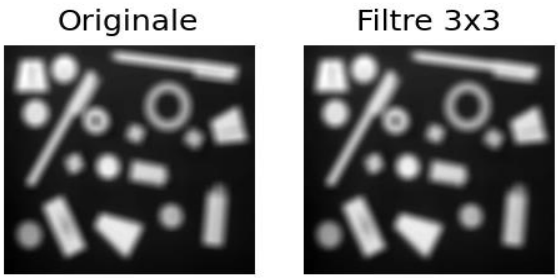
Code python :

```
####  Filtre moyenneur #####
#Q1.1 et Q1.2
import cv2
import numpy as np
import matplotlib.pyplot as plt

images = ["hhrec.bmp", "lenam.bmp", "objectm.bmp"]
filter_sizes = [3, 5, 7, 11]

for img_name in images:
    img = cv2.imread(img_name, cv2.IMREAD_GRAYSCALE)

    plt.figure(figsize=(10, 5))
    plt.subplot(1, len(filter_sizes) + 1, 1)
    plt.imshow(img, cmap="gray")
    plt.title("Originale")
    plt.axis("off")

    for i, k in enumerate(filter_sizes):
        img_filtered = cv2.blur(img, (k, k))
        plt.subplot(1, len(filter_sizes) + 1, i + 2)
        plt.imshow(img_filtered, cmap="gray")
        plt.title(f"Filtre {k}x{k}")
        plt.axis("off")
        cv2.imwrite(f"{img_name.split('.')[0]}_filtered_{k}x{k}.bmp",
img_filtered)

    plt.show()
```

1. Appliquez le filtre Moyenneur 3*3 aux images de travail, afficher et comparer

Résultat:

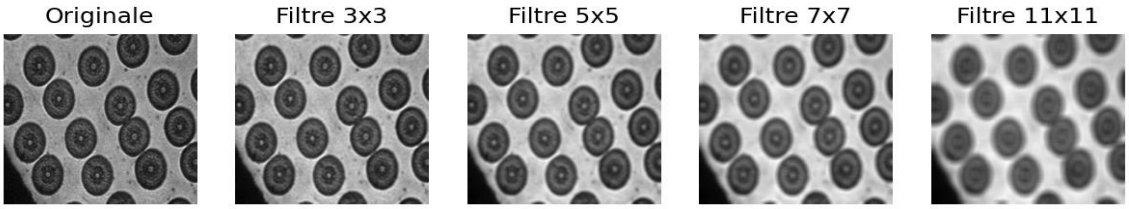




- Les images sont légèrement lissées, mais les détails sont encore bien visibles.

2. Appliquez une fenêtre 5*5, 7*7 et 11*11, afficher et Discuter l'effet de l'augmentation de la taille du filtre sur ces images

Résultat:



- **En utilisant 5×5 :** Le bruit est plus réduit, mais un léger flou commence à apparaître.
- **En utilisant 7×7 et 11×11 :** Les images deviennent de plus en plus floue, avec une forte perte de détails.

➢ On peut déduire qu'une taille de filtre plus grande réduit mieux le bruit, mais rend l'image plus floue.

2 Filtre médian

Le filtre médian remplace chaque pixel par la médiane de ses voisins dans une fenêtre donnée.

Cela permet de réduire le bruit tout en conservant les contours**.**

1.Appliquez le filtre médian 3*3 aux images f1, f2 puis f1 + f2 et comparer les résultats

Code python :

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

f1 = np.array([
    [1, 1, 3, 1, 2, 1, 1],
    [3, 0, 3, 1, 3, 2, 0],
    [1, 3, 6, 5, 9, 6, 0],
    [0, 3, 4, 5, 7, 8, 2],
    [2, 0, 1, 0, 4, 6, 0],
    [0, 2, 1, 2, 7, 4, 0]
])

f2 = np.array([
    [0, 2, 0, 3, 5, 7, 0],
    [1, 2, 1, 1, 6, 6, 0],
    [0, 1, 0, 0, 3, 1, 0],
    [2, 0, 1, 3, 2, 1, 0],
    [2, 1, 3, 1, 1, 3, 0],
    [3, 2, 1, 2, 3, 1, 0]
])

f_sum = f1 + f2
```

```
f1_median = cv2.medianBlur(f1.astype(np.uint8), 3)
f2_median = cv2.medianBlur(f2.astype(np.uint8), 3)
f_sum_median = cv2.medianBlur(f_sum.astype(np.uint8), 3)

print("Image f1 filtrée :\n", f1_median)
print("Image f2 filtrée :\n", f2_median)
print("Image f1 + f2 filtrée :\n", f_sum_median)
```

Résultat:

```
Image f1 filtrée :
 [[1 1 1 2 1 1 1]
 [1 3 3 3 2 2 1]
 [1 3 3 5 5 3 2]
 [1 2 3 5 6 6 2]
 [0 1 2 4 5 4 2]
 [0 1 1 2 4 4 0]]
Image f2 filtrée :
 [[1 1 2 3 5 5 0]
 [1 1 1 1 3 3 0]
 [1 1 1 1 2 1 0]
 [1 1 1 1 1 1 0]
 [2 2 1 2 2 1 0]
 [2 2 2 2 2 1 0]]
Image f1 + f2 filtrée :
 [[2 3 3 4 7 7 1]
 [2 3 4 5 7 7 1]
 [2 4 4 6 8 8 2]
 [2 4 4 5 8 7 2]
 [3 3 4 5 8 5 2]
 [3 3 4 4 5 5 0]]
PS C:\Users\user\Desktop\SDA Master\S2\TDM\TP2TNI>
```

Comparaison des résultats :

Le filtre médian 3x3 lisse les matrices en réduisant les variations locales. Sur **f1** et **f2**, il atténue les valeurs extrêmes tout en préservant la structure. Sur **f1 + f2**, l'effet est plus marqué, offrant un résultat plus homogène

2. Appliquez les filtres médian 3*3, 5*5, 7*7, 9*9 et 11*11 à l'image lenam et commenter les résultats.

Code python:

```
img = cv2.imread("lenam.bmp", cv2.IMREAD_GRAYSCALE)

filter_sizes = [3, 5, 7, 9, 11]
```

```
plt.figure(figsize=(12, 6))
plt.subplot(1, len(filter_sizes) + 1, 1)
plt.imshow(img, cmap="gray")
plt.title("Originale")
plt.axis("off")

for i, k in enumerate(filter_sizes):
    img_filtered = cv2.medianBlur(img, k)

    plt.subplot(1, len(filter_sizes) + 1, i + 2)
    plt.imshow(img_filtered, cmap="gray")
    plt.title(f"Filtre {k}x{k}")
    plt.axis("off")

    cv2.imwrite(f"lenam_filtered_{k}x{k}.bmp", img_filtered)

plt.show()
plt.close('all')
```

Résultat:

- **Filtre 3×3** : Réduction du bruit avec très peu de perte de détails**.**
- **Filtre 5×5** : Atténuation plus forte du bruit, mais début de lissage des contours.
- **Filtre 7×7 et 9×9** : Image plus lisse, mais les détails fins disparaissent progressivement**.**
- **Filtre 11×11** : Image fortement lissée, perte notable des détails et texture plus homogène**.**

Conclusion

- Le filtre médian est très efficace pour éliminer le bruit impulsionnel tout en préservant les contours.
- Un filtre trop grand peut excessivement lisser l'image et supprimer les détails utiles**.**

3 Débruitage des images

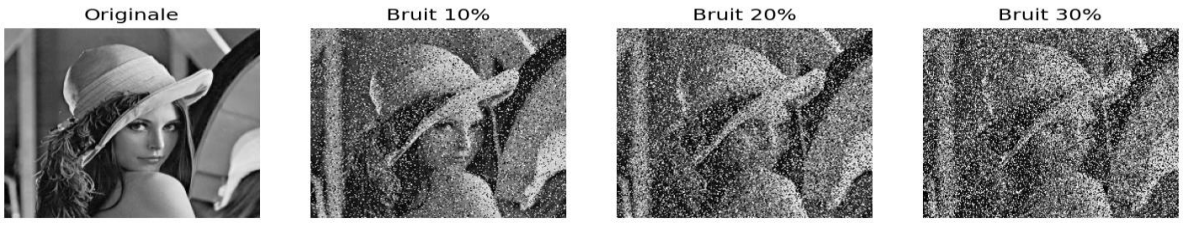
1. ajouter un bruit poivre et sel à lenam et chosissez 3 pourcentages de bruit dans l'image

Code python :

```
import numpy as np
```

```
import cv2
import random
import matplotlib.pyplot as plt

def add_noise(image, percentage, noise_type):

    noisy_image = image.copy()
    num_pixels = int(image.size * percentage / 100)

    for _ in range(num_pixels):
        x, y = random.randint(0, image.shape[1]-1), random.randint(0,
image.shape[0]-1)
        if noise_type == "poivre":
            noisy_image[y, x] = 0
        elif noise_type == "sel":
            noisy_image[y, x] = 255
        elif noise_type == "poivre_et_sel":
            noisy_image[y, x] = 0 if random.random() < 0.5 else 255

    return noisy_image

img = cv2.imread("lenam.bmp", cv2.IMREAD_GRAYSCALE)
if img is None:
    raise FileNotFoundError("L'image 'lenam.bmp' est introuvable.")

# 1 Ajout du bruit poivre et sel avec différents niveaux (10%, 20%, 30%)
noise_levels = [10, 20, 30]
noisy_images = [add_noise(img, p, "poivre_et_sel") for p in noise_levels]
```

Résultat:



- Plus le pourcentage de bruit augmente, plus l'image devient dégradée avec des points noirs et blancs répartis aléatoirement.

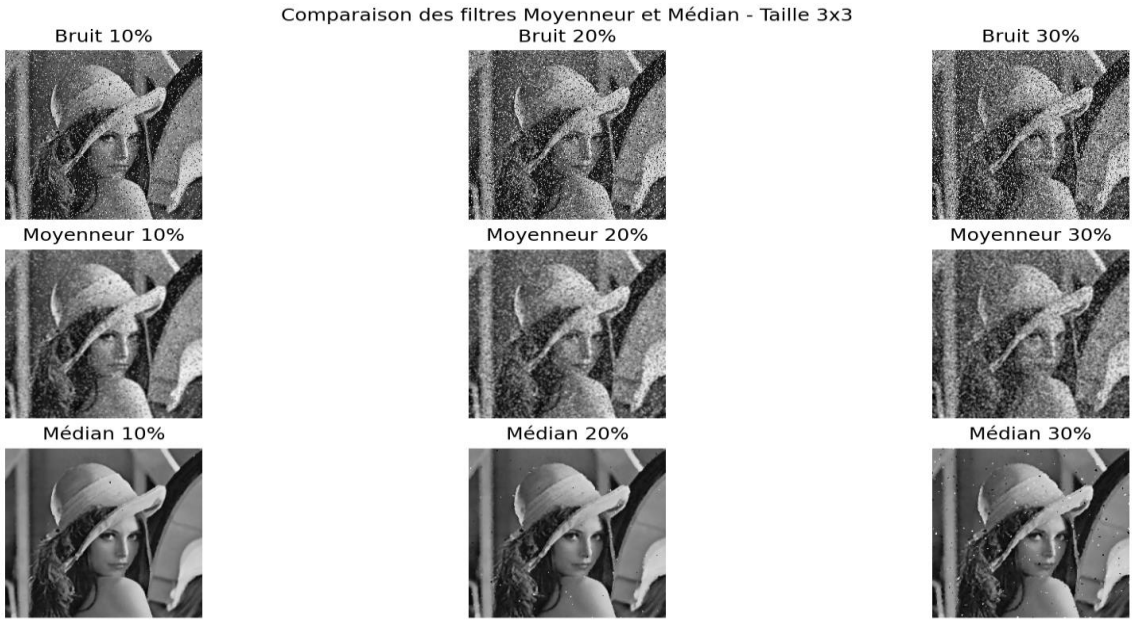
- À 30 %, les détails deviennent difficilement visibles, ce qui complique le débruitage.

2. Utilisez les filtres moyenneur et median de même taille pour essayer d'enlever le bruit en adaptant les fenêtres, Que remarquez-vous ?

Code python :

```
# 2 Application des filtres moyenneur et médian
filter_sizes = [3, 5, 7, 11]  # Tailles des fenêtres de filtrage

for size in filter_sizes:
    plt.figure(figsize=(12, 8))
    plt.suptitle(f"Comparaison des filtres Moyenneur et Médian - Taille
{size}x{size}")

    for i, noisy_img in enumerate(noisy_images):
        # Filtre Moyenneur
        avg_filtered = cv2.blur(noisy_img, (size, size))
        # Filtre Médian
        median_filtered = cv2.medianBlur(noisy_img, size)

        plt.subplot(3, 3, i+1)
        plt.imshow(noisy_img, cmap="gray")
        plt.title(f"Bruit {noise_levels[i]}%")
        plt.axis("off")

        plt.subplot(3, 3, i+4)
        plt.imshow(avg_filtered, cmap="gray")
        plt.title(f"Moyenneur {noise_levels[i]}%")
        plt.axis("off")

        plt.subplot(3, 3, i+7)
        plt.imshow(median_filtered, cmap="gray")
        plt.title(f"Médian {noise_levels[i]}%")
        plt.axis("off")

    plt.tight_layout()
    plt.show()
```

Résultat:


Comparaison des filtres Moyenneur et Médian - Taille 3x3
Bruit 10%
Bruit 20%
Bruit 30%
Moyenneur 10%
Moyenneur 20%
Moyenneur 30%
Médian 10%
Médian 20%
Médian 30%

Comparaison des filtres Moyenneur et Médian - Taille 5x5

Bruit 10%

Bruit 20%

Bruit 30%

Moyenneur 10%

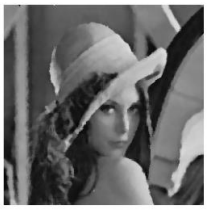
Moyenneur 20%

Moyenneur 30%

Médian 10%

Médian 20%

Médian 30%



Comparaison des filtres Moyenneur et Médian - Taille 7x7

Bruit 10%

Bruit 20%

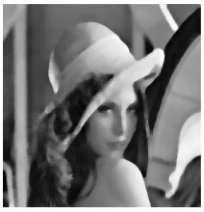
Bruit 30%

Moyenneur 10%

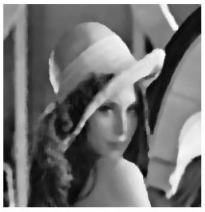
Moyenneur 20%

Moyenneur 30%

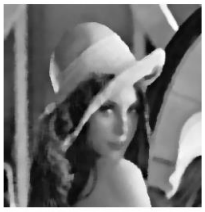
Médian 10%

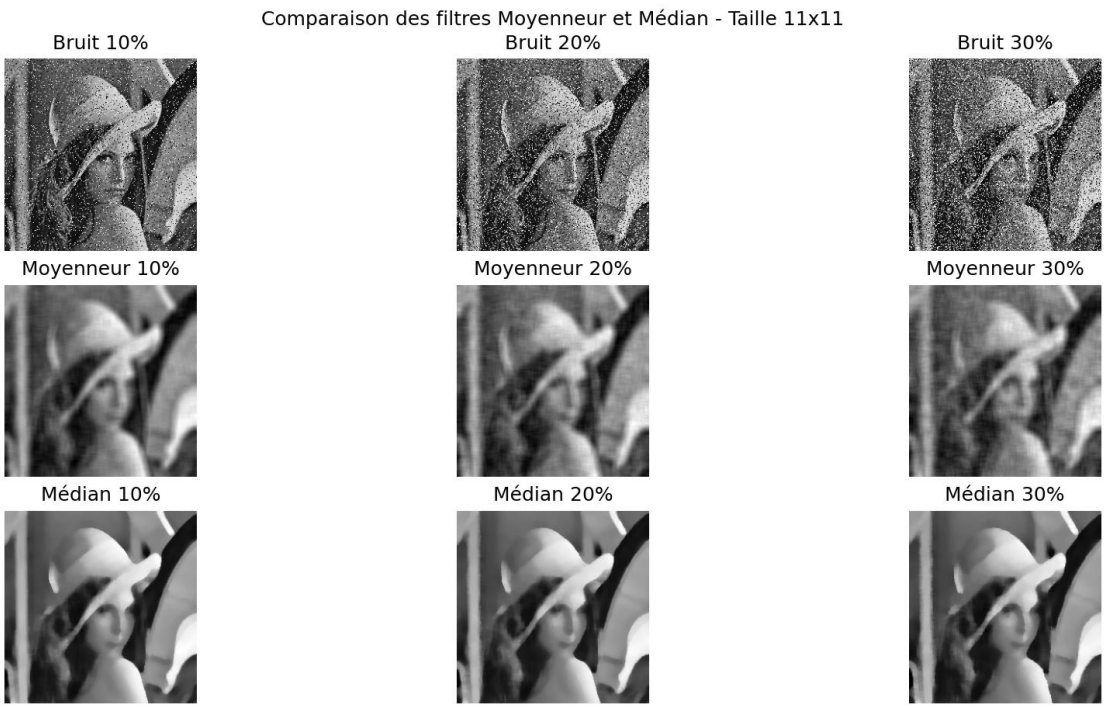


Médian 20%



Médian 30%





Remarque :

Le filtre moyenneur : atténue le bruit mais introduit un effet de flou, surtout avec des tailles de fenêtre élevées.

Le filtre médian : élimine mieux le bruit tout en préservant les contours, ce qui le rend plus efficace pour les images avec du bruit poivre et sel.

3.Quelle méthode enlève le mieux le bruit tout en préservant le contenu de l'image ?

Le filtre médian : est plus efficace car il supprime le bruit poivre et sel sans trop flouter l'image**.**

Le filtre moyenneur : réduit le bruit, mais il rend l'image floue et moins nette**.**

- Le filtre médian est le plus adapté pour ce type de bruit

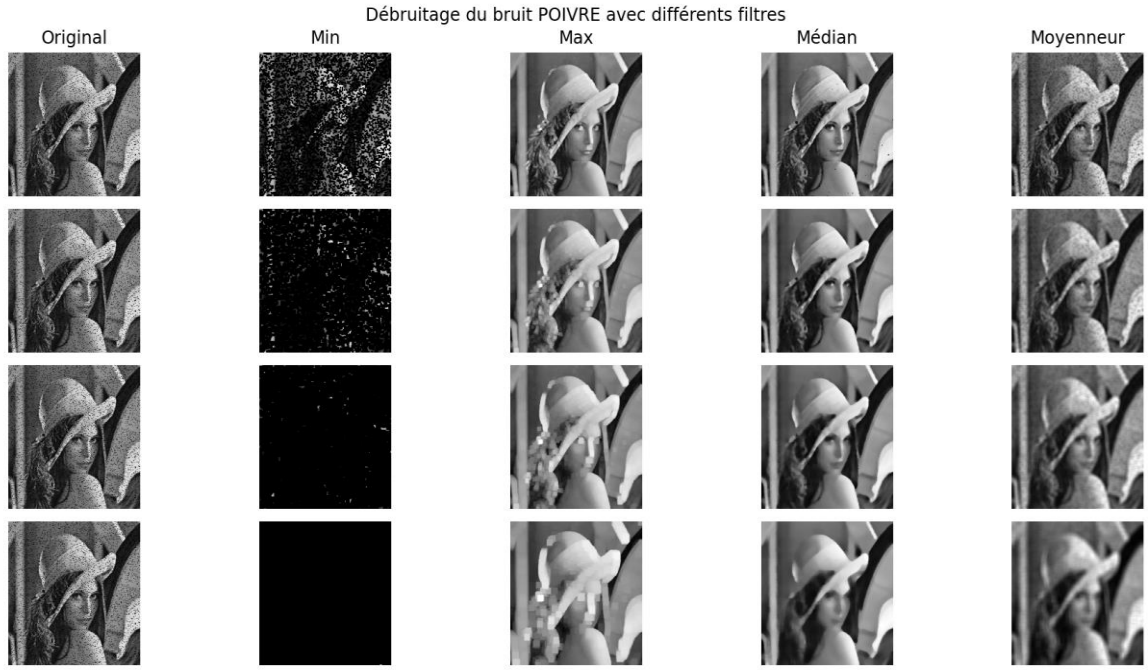
4. Refaire la même chose avec un bruit poivre, puis un bruit sel et comparer l'efficacité des filtre min, max, médian et moyenneur en variant les tailles des fenêtres

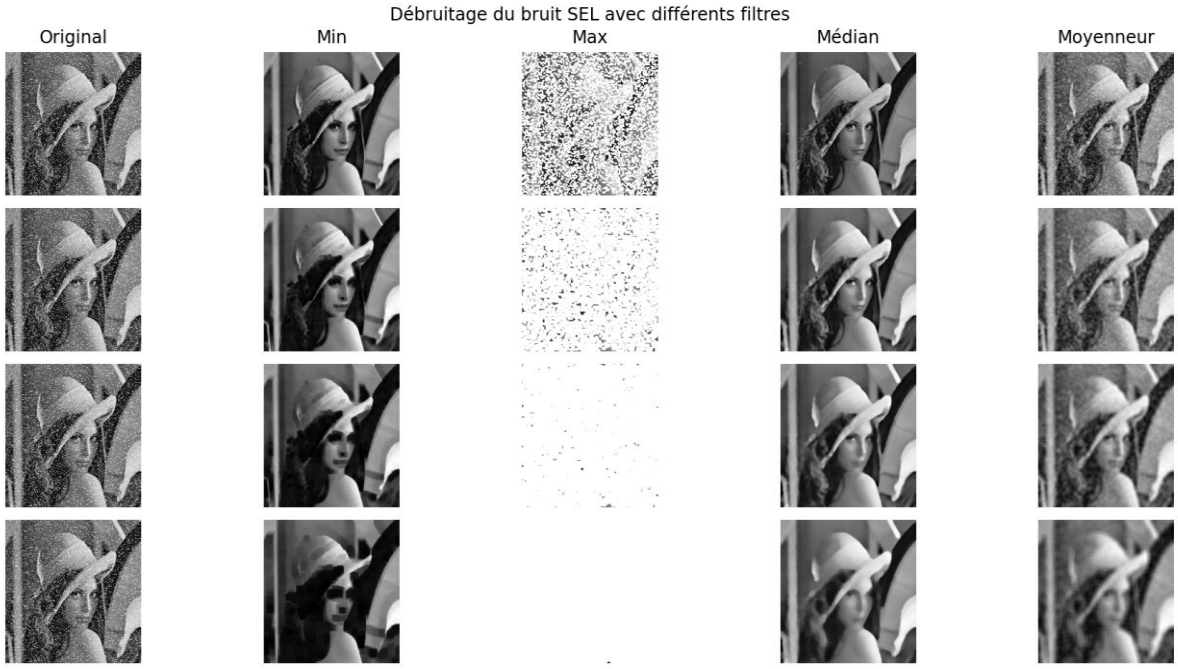
Code python :

```
# 3 Ajout du bruit Poivre et Sel séparément
noise_type_list = ["poivre", "sel"]
for noise_type in noise_type_list:
```

```
    noisy_image = add_noise(img, 10, noise_type)  # Fixe à 10% pour la
comparaison

    plt.figure(figsize=(12, 8))
    plt.suptitle(f"Débruitage du bruit {noise_type.upper()} avec différents
filtres")

    for j, size in enumerate(filter_sizes):
        min_filter = cv2.erode(noisy_image, np.ones((size, size)))  # Filtre
Min
        max_filter = cv2.dilate(noisy_image, np.ones((size, size)))  # Filtre
Max
        median_filter = cv2.medianBlur(noisy_image, size)  # Filtre Médian
        avg_filter = cv2.blur(noisy_image, (size, size))  # Filtre Moyenneur

        titles = ["Original", "Min", "Max", "Médian", "Moyenneur"]
        images = [noisy_image, min_filter, max_filter, median_filter,
avg_filter]

        for i in range(5):
            plt.subplot(len(filter_sizes), 5, j*5 + i + 1)
            plt.imshow(images[i], cmap="gray")
            if j == 0:
                plt.title(titles[i])
            plt.axis("off")

    plt.tight_layout()
    plt.show()
```

Résultat:





Remarque :

Min : Remplace chaque pixel par le minimum de la fenêtre (efficace contre le bruit sel).

Max : Remplace chaque pixel par le maximum de la fenêtre (efficace contre le bruit poivre).

Médian : Bon contre les deux.

Moyenneur : Lisse l'image mais floute les détails.