TP2 : Filtrage et Restauration – Chloé Court

2 -

Rotation avec la méthode du plus proche voisin

Rotation avec la méthode bilinéaire

On remarque que l'image pour laquelle on a utilisé la méthode du plus proche voisin est plus granuleuse que celle où on a utilisé la méthode bininéaire. De plus, au niveau du bas du chapeau on remarque que les pixels sont visibles avec la méthode du plus proche voisin, ce qui n'est pas le cas pour la méthode bilinéaire. Ainsi la méthode du plus proche voisin est moins précise notamment sur les hautes fréquences (lignes et bordures). La méthode bilinéaire paraît plus floue.

8 rotations avec la méthode du plus proche voisin

8 rotations avec la méthode bilinéaire

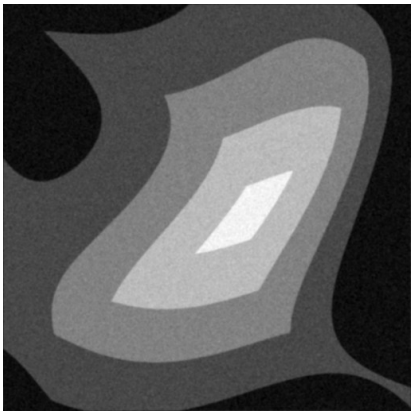
On remarque que lorsqu'on applique 8 fois la rotation on perd les informations de l'image sur les bords. Entre l'image et le noir on observe également des dents de scie. De plus, les pixels sont visibles et il y a une impression de flou pour chacun d'entre eux. En revanche, pour les 8 rotations avec la méthode bilinéaire l'image est floue mais les pixels ne sont pas apparents. Il n'y a pas de d'effet dents de scie au niveau des bords.

Lorsque l'on applique la rotation avec un facteur de zoom de 1/2, on a cette fois aucune perte d'informations au niveau des bords de l'image. Cependant, les pixels sont visibles, et ce même avec la méthode bilinéaire, et on observe l'apparition de courbes au niveau du chapeau qui n'étaient pas présentes dans l'image originale. Cela est dû au fait que lors du zoom on a fait une moyenne des pixels sans appliquer le filtre passe-bas. Les courbes sont donc le résultat d'un repliement spectral. Pour atténuer l'effet constaté on pourrait ainsi appliquer un filtre passe-bas.

3 -

Le paramètre s de la fonction get_gau_ker correspond à l'écart-type de la gaussienne qui génère le noyau. Cela permet de déterminer le support de la gaussienne, qui doit être fini, et donc la taille du noyau.

Image de la pyramide avant filtrage et bruitage



Pyramide bruitée avec br=10 et filtrée linéairement avec un écart-type de la gaussienne de 1

Pyramide bruitée avec br=10 et filtrée linéairement avec un écart-type de la gaussienne de 5

Ainsi, on remarque que la quantité de bruit résiduel se traduit par la variation de niveaux de gris dans une zone de couleur donnée. Le filtre linéaire prend la moyenne de ces niveaux de gris ce qui a pour conséquence de réduire le bruit mais donc aussi de flouter (moyenne au niveau des bords).

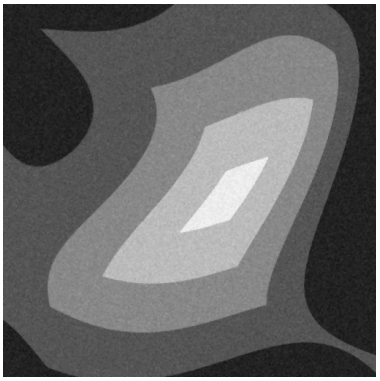


Image de la pyramide bruitée avec br=10 et filtrage médian avec r=1

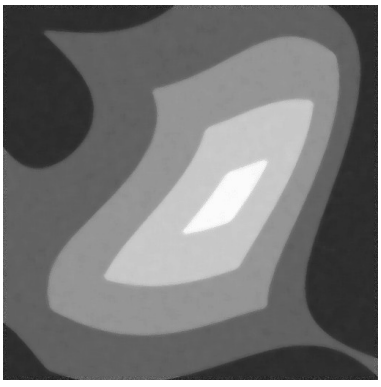


Image de la pyramide bruitée avec br=10 et filtrage médian avec r=5

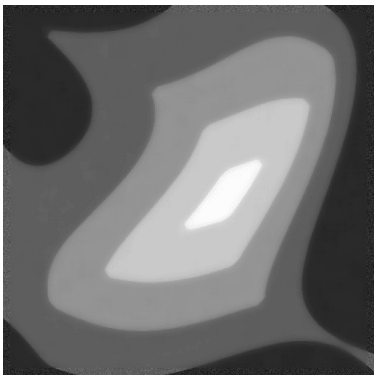


Image de la pyramide bruitée avec br=10 et filtrage médian avec r=10

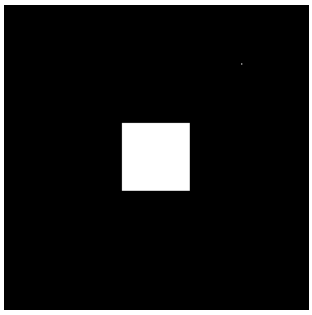
Le filtrage médian réduit le bruit de façon équivalente au filtrage linéaire sur du bruit additif. Or, on observe que le fait de prendre la médiane des pixels voisins plutôt que la moyenne permet de

garder l'information des bords entre les niveaux de gris. Pour autant, on voit bien avec r=10 que l'image est bel et bien débruitée. On remarque que le filtrage médian a arrondi les bords.

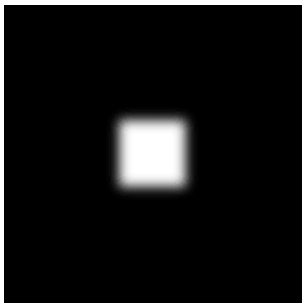
Pyramide bruitée filtrée linéairement avec un écart-type de la gaussienne de 5

Image de la pyramide et filtrage médian avec r=5

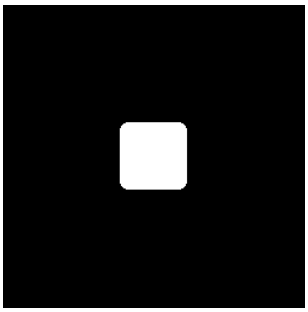
On constate que le filtre médian est plus efficace pour la réduction du bruit que le filtre linéaire sur du bruit impulsionnel. On a le même résultat que précédemment au niveau des bords, à savoir que le filtre médian les conserve, à l'inverse du filtre linéaire.



Carré original avec le point blanc



Carré filtré linéairement



Carré filtre médian

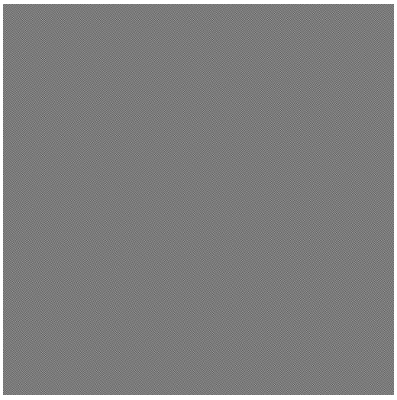
Les deux filtres ne permettent pas de conserver l'information du point blanc en haut à gauche. Le filtre linéaire floute encore les bords et le filtre médian n'est pas flou mais arrondi les bords. Cela est cohérent avec nos précédentes interprétations.

4 -

Image filtrée

Image filtrée puis filtrée inversement

L’image filtrée est floutée tandis que l’image filtrée puis filtrée inversement est identique à l’image d’origine. On n’observe donc pas visuellement de perte d’information.



Lorsque l’on rajoute du bruit sur l’image filtrée, toutes les informations sur l’image sont perdues lors de l’application du filtre inverse.

Pour déterminer le noyau de convolution qu’a subi l’image carre_flou, on peut s’appuyer sur le point lumineux en haut à droite de l’image. En effet, celui-ci s’apparente à un Dirac en un point donné. Ainsi, sa convolution avec le noyau de convolution va être le noyau de convolution évalué en ce point même. Il suffit donc de s’interesser au noyau de convolution au niveau du pixel du point lumineux pour pouvoir déterminer le noyau de convolution.

Image restaurée avec lambda = 0,5, bruit sous estimé

Image restaurée avec lambda = 1, amélioration par rapport à lambda =0,5

Image restaurée avec lambda = 2, bruit sur estimé

Image restaurée avec lambda = 3, bruit sur estimé et image détériorée par rapport à lambda =2

Ainsi, en prenant un écart type de 2 pour le noyau, on observe en modifiant le lambda que l'image s'améliore d'abord en changeant le lambda (le bruit est d'abord sous-estimé) jusqu'à environ 1. Ensuite, l'image se détériore en continuant d'augmenter lambda. Le bruit est alors sur-estimé et on observe un effet flou. Lorsque lambda vaut 0, on constate que l'on obtient la même chose qu'avec le filtre inverse, ce qui est cohérent. Aucune information n'est conservée.

5.1 -

```
im=skio.imread('images/carre_orig.tif')
im=noise(im, 5)

i = 1
while(var_image(filtre_lineaire(im,get_cst_ker(i)),20,20,40,40) >= var_image(median_filter(im,
typ=2, r=4),20,20,40,40)):
    i +=1
if(abs(var_image(filtre_lineaire(im,get_cst_ker(i)),20,20,40,40) - var_image(median_filter(im,
typ=2, r=4),20,20,40,40)) > abs(var_image(filtre_lineaire(im,get_cst_ker(i-1)),0,0,1,1) -
var_image(median_filter(im, typ=2, r=4),0,0,1,1))):
    print(i-1)
else :
    print(i)
```

Voici le code qui permet de trouver la taille du noyau constant réduisant le bruit de manière équivalente au filtrage médian. En effet, pour i=1, on sait que la variance de l'image filtrée linéairement est supérieure à celle filtrée avec le filtre médian, puisque l'image est plus bruitée. On utilise la fonction var_image sur un patch significatif (largeur et hauteur de 40 pixels). Ainsi, on augmente progressivement i jusqu'à ce que l'image soit moins bruitée avec le filtre linéaire qu'avec le filtre médian. Une fois que l'on a ce i, on vérifie quelle valeur entre i et i-1 permet d'avoir un bruit le plus proche de celui obtenu avec le filtre médian.

On obtient avec ce code i=6, ce qui est cohérent visuellement :

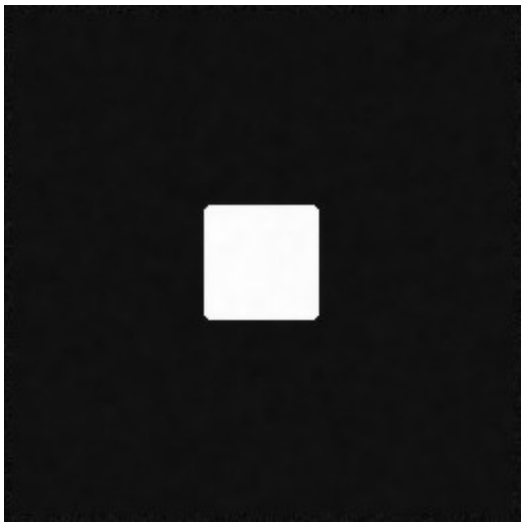


Image filtrée avec le filtre médian circulaire de rayon 4

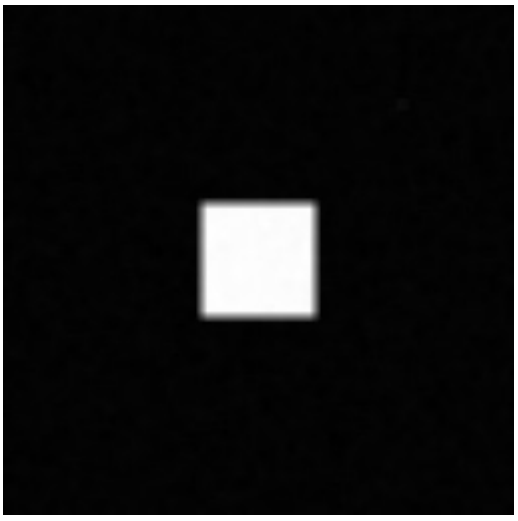


Image filtrée avec le filtre linéaire de noyau constant de taille 6

5.2 -

```
def modified_wiener(im, K):
    fft2=np.fft.fft2
    ifft2=np.fft.ifft2
    (ty,tx)=im.shape
    (yK,xK)=K.shape
    KK=np.zeros((ty,tx))
    KK[:yK,:xK]=K
    x2=tx/2
    y2=ty/2

    fX=np.concatenate((np.arange(0,x2+0.99),np.arange(-x2+1,-0.1)))
    fY=np.concatenate((np.arange(0,y2+0.99),np.arange(-y2+1,-0.1)))
    fX=np.ones((ty,1))@fX.reshape((1,-1))
    fY=fY.reshape((-1,1))@np.ones((1,tx))
    fX=fX/tx
    fY=fY/ty

    #tranformee de Fourier de l'image degradeee
    g=fft2(im)
    #transformee de Fourier du noyau
    k=fft2(KK)

    nb_pixels = im.size
    #fonction de mutilplication
    mul=np.conj(k)/(abs(k)**2+nb_pixels * np.var(im)/(g**2))
    #filtrage de wiener
    fout=g*mul

    # on effectue une translation pour une raison technique
    mm=np.zeros((ty,tx))
    y2=int(np.round(yK/2-0.5))
    x2=int(np.round(xK/2-0.5))
    mm[y2,x2]=1
    out=np.real(ifft2(fout*(fft2(mm))))
    return out
```

Voici le code permettant d'utiliser le spectre de l'image dégradée. Ainsi, on obtient les images suivantes respectivement avec weiner et modified wiener :

Restauration wiener

Restauration modified wiener

On constate que les deux approches n’aboutissent pas à la même restauration. En effet, la première approche est plus bruitée mais conserve mieux la forme tandis que la seconde est moins bruitée mais moins précise géométriquement.