

TP Filtrage et Restauration

2. Transformation Géométrique

Le méthode plus proche voisin a une résolution avec plus des erreurs pour sa nature de chercher seulement le pixel plus proche de l'image originale, il peut tomber en erreurs en donnant une image pixellisée, au contraire la méthode bilinéaire peut générer des images plus doux avec transitions graduelles



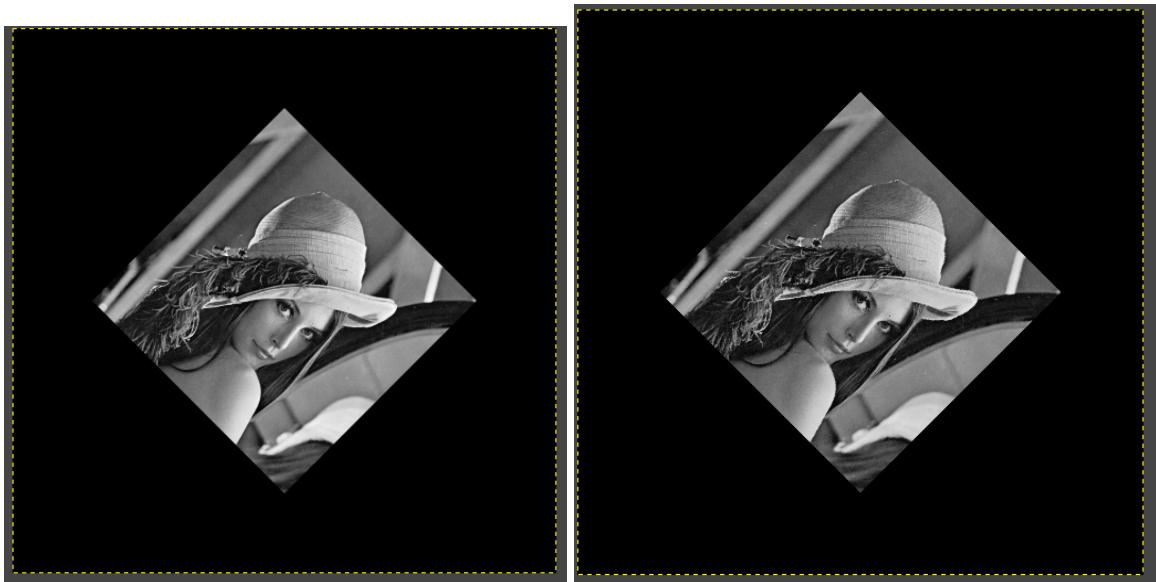
8 Rotations

On peut constater que dans l'image avec le méthode de plus proche voisin on voit l'image plus timide, mais très pixellisé aussi, dans l'image avec le méthode bilinéaire, l'image est plus doux, cependant elle a perdu de qualité pendant les rotations



Zoom

Dans ce cas, on constate la même que dans les antérieurs cas, l'image avec le méthode bilinéaire est plus doux, est avec plus proche voisin plus pixellisé, pour atténuer cet effet on pourrait mettre un filtre passe bas



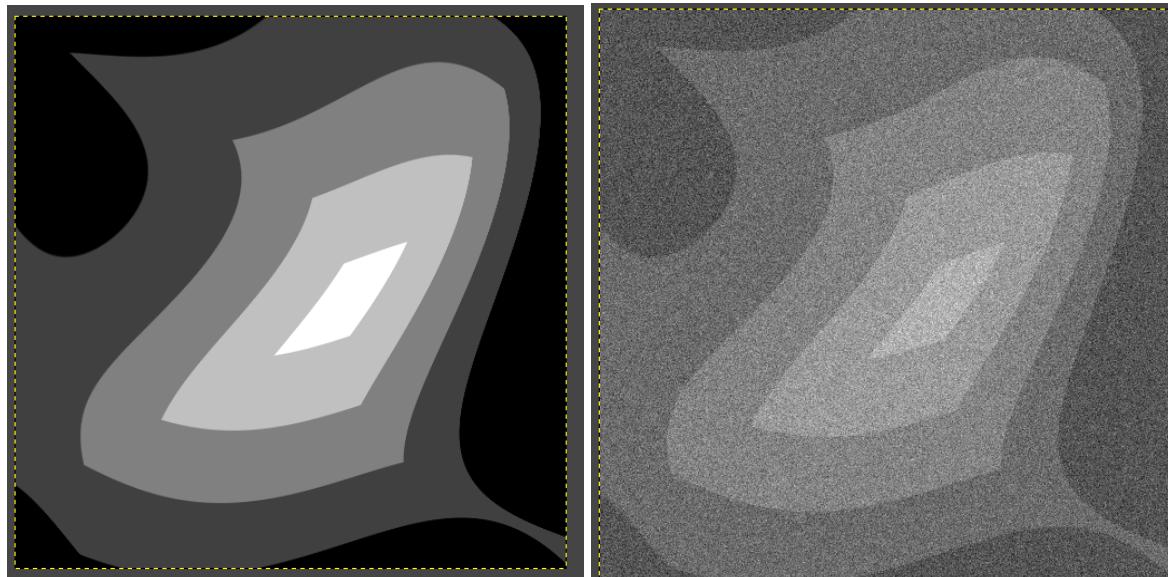


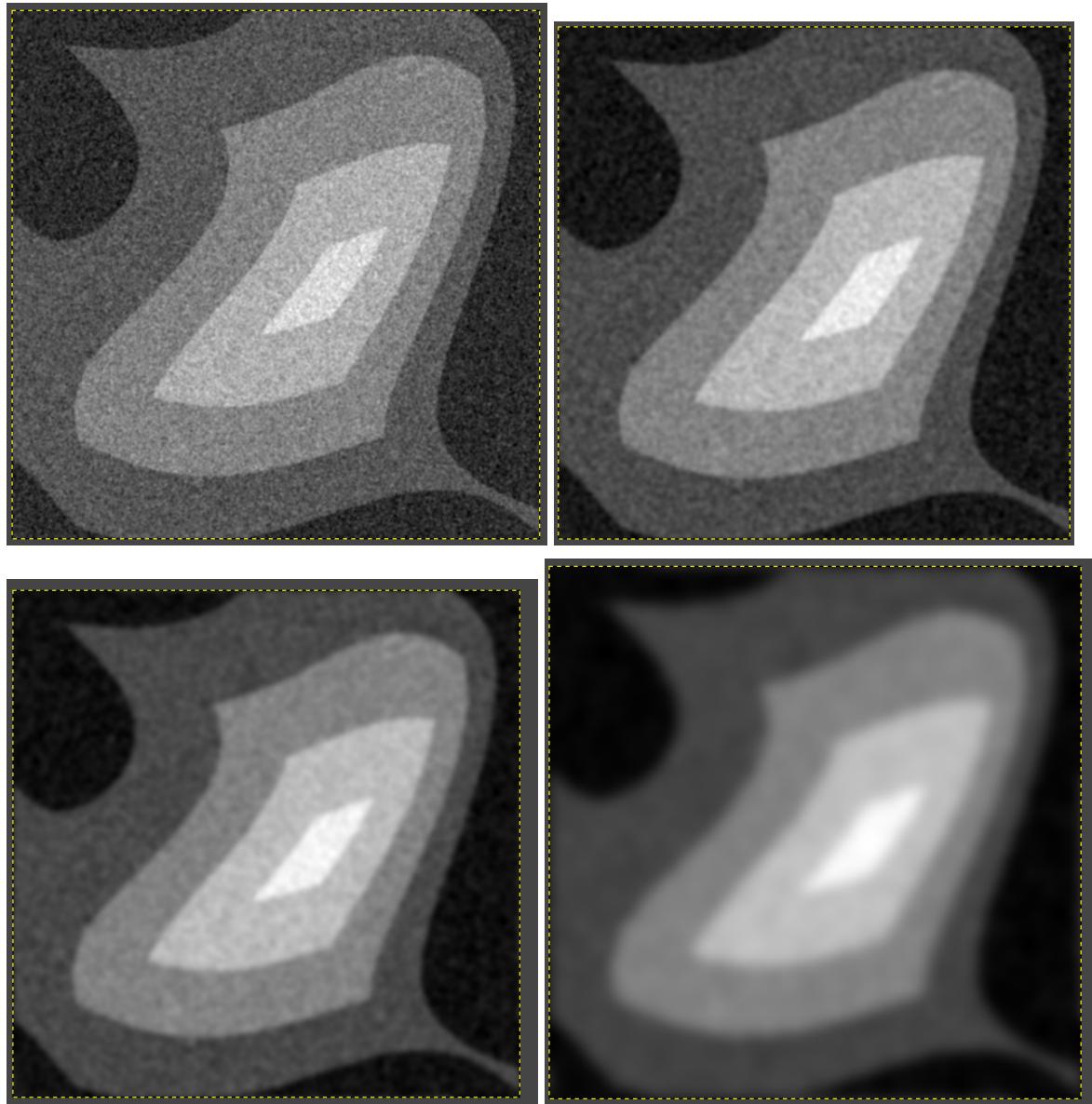
3. Filtrage Linéaire et médian

La taille de noyau depend de parametre **s**, alors si le paramètre s augmente, la taille augmente aussi pour la corrélation linéaire, plus grand s, plus grande la taille de la cloche gaussienne, alors il va réduire le bruit, pendant on perd de qualité

On trouve que plus grand la taille du filtre, c'est mineur le bruit résiduel, c'est logique, parce que quand on augmente le size, on va faire la convolution de la région avec les autres pixels dans la grille, c'est à dire que le bruit va se réduire, et la qualité de l' image aussi

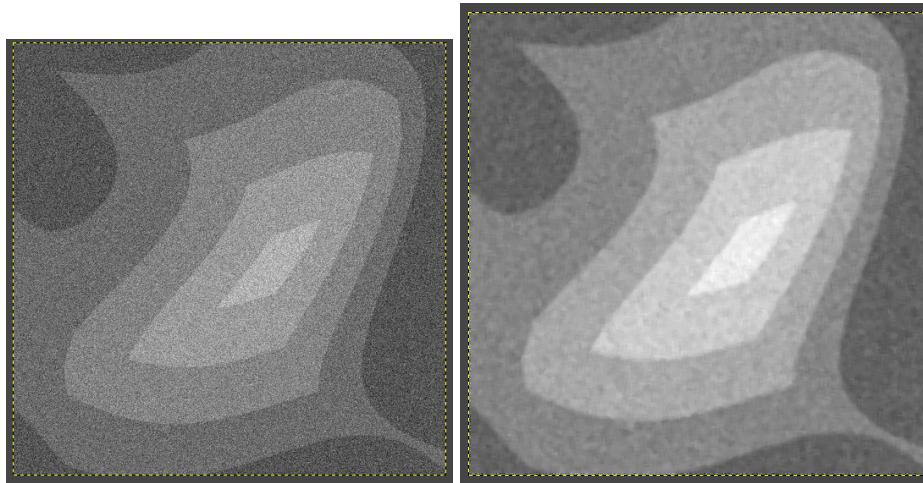
```
Image with gaussian filter size: 1 has a residual noise of : 135.35021860601594
Image with gaussian filter size: 3 has a residual noise of : 15.963327353027788
Image with gaussian filter size: 5 has a residual noise of : 7.171893276138476
Image with gaussian filter size: 7 has a residual noise of : 5.689162392467144
Image with gaussian filter size: 9 has a residual noise of : 6.062798534202648
```





On peut trouver qu'avec le filtre médian l'image a une plus de variance, c'est à dire que a plus de bruit résiduel, cependant, l'image avec le filtre linéaire, a plus de similitude a l'image original que l' image avec filtre médian, alors, dans ce cas le filtrage linéaire est très mieux

101.12897940939376



Dans cette image, on peut trouver que la meilleure option est utilisée le filtrage médian, bien que le gaussian montre une meilleure variation, l'image avec le filtrage médian montre une meilleur résultat en rapport à la qualité

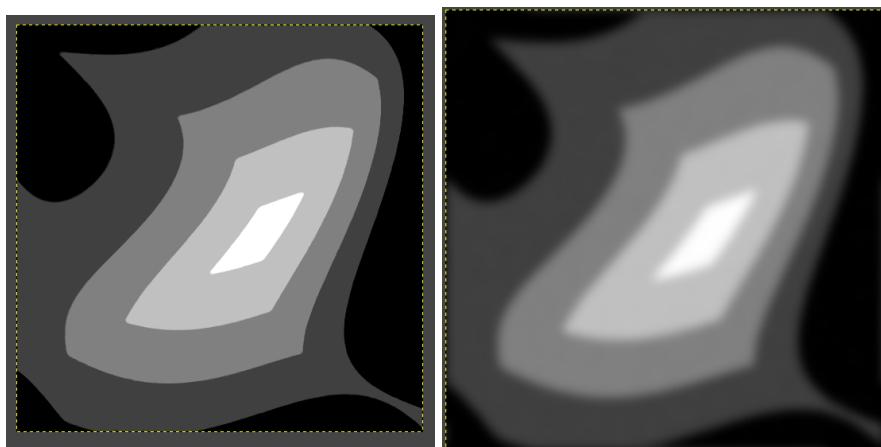
Image with gaussian filter size: 2 has a residual noise of : 0.6922831901225037

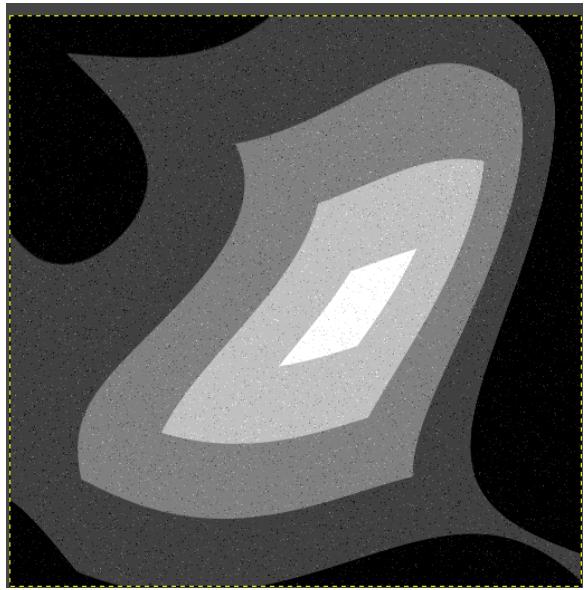
Image with gaussian filter size: 4 has a residual noise of : 0.12880298952407812

Image with gaussian filter size: 6 has a residual noise of : 0.03566491227296722

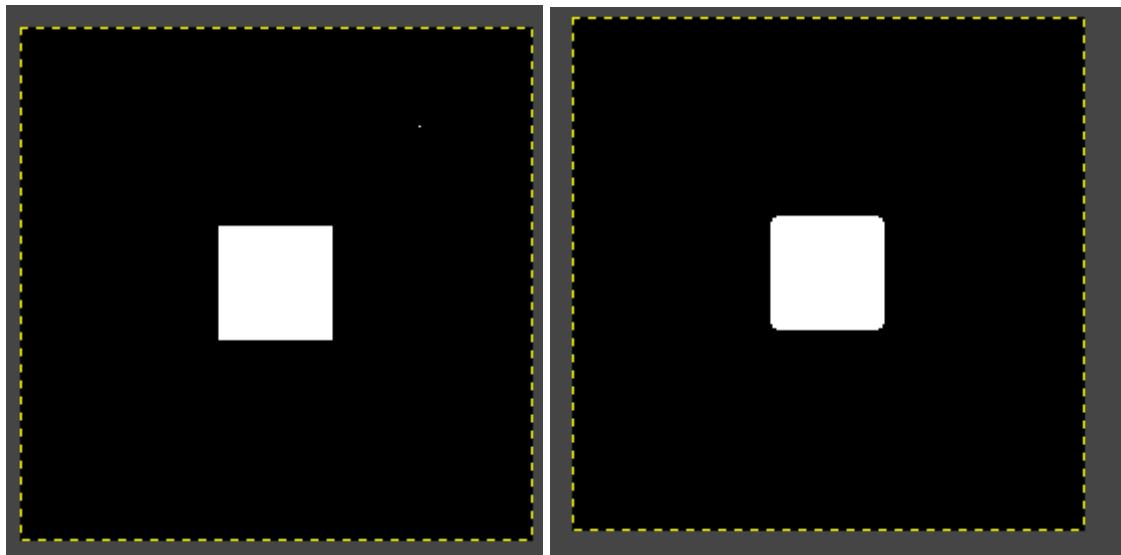
Image with gaussian filter size: 8 has a residual noise of : 0.013993607153015626

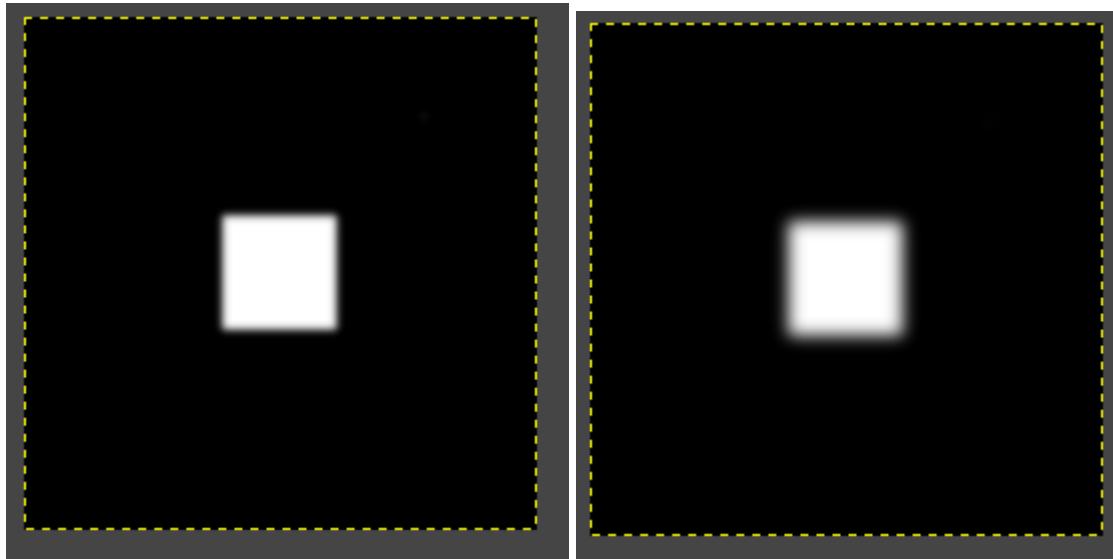
Image with median filter size: 3 has a residual noise of : 1.8980733905762732





Dans cette image, dans les deux cas, on perd le point lumineux, cependant dans l'image avec le filtrage médian on conserve la qualité du carré, mais on perd aussi un peu des bords dans le carré, ce qui ne se passe pas dans le filtre gaussien, où on peut voir le carré très pixelé



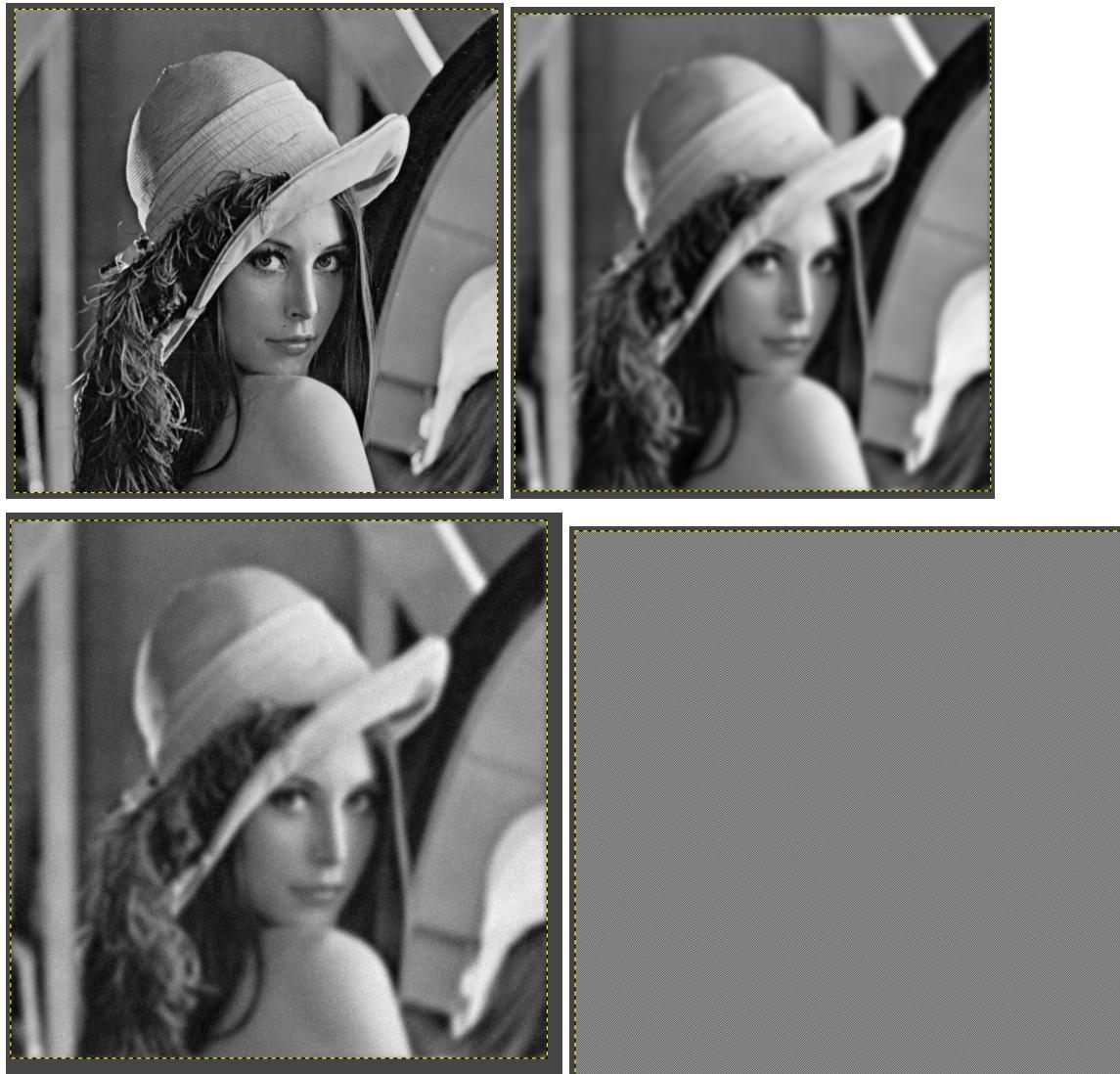


4. Restauration

Après appliquer un filtre linéaire, et faire l'inverse on peut constater qu'on a la même image, parce que effectivement, on a des pixels, applique une fonction mathématique, et après on fait le contraire à cette fonction, on aura alors le même pixel



Quand on ajoute de bruit, à l'image avec le filtre, on va constater que l'inverse sera totalement inutile, parce que, avec la même raison que dans la question antérieure, on a un formule mathématique, alors, si on fait des graduations à l'image, c'est à dire, on change les pixels, de l'image avec le filtre, le résultat, sera totalement différent à l'image original



Dans le image carre_flou, on peut noter un T dans la même position qu'il y a un point blanc dans carre_orig, alors, on peut déduire, que avec ce point blanc on peut générer le T dans le image flou, pourtant le noyau qu'on applique est:

$\begin{bmatrix} 0,0,1 \\ 1,1,1 \\ 0,0,1 \end{bmatrix}$

On arrive après augmenter lambda un bon résultat, après ça il se réduit si on continue augmentant lambda, parce qu'il doit être proportionnel à le bruit qui a l'image

5. Applications

```
im=skio.imread('images/carre_orig.tif')
im_noise = noise(im, 20)
long = 100
```

```

median_var = var_image(median_filter(im_noise, typ=2, r=4), 0, 0, 50,
50)

for i in range(1, 100):
    linear_var = var_image(filtre_lineaire(im, get_cst_ker(i)),
0, 0, 50, 50)
    if (abs(linear_var-median_var)/median_var) < long:
        long = abs(linear_var-median_var)/median_var
        index = i

```

```

def wiener(im,K,var_b=0):
    """effectue un filtrage de wiener de l'image im par le filtre K.
    lamb=0 donne le filtre inverse
    on rappelle que le filtre de Wiener est une tentaive d'inversion
du noyau K
    avec une regularisation qui permet de ne pas trop augmenter le
bruit.

    """
    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

    w2=fX**2+fY**2
    w=w2**0.5

    #tranformee de Fourier de l'image degradee
    g=fft2(im)
    #densite spectrale de l'image
    sigma_s=g**2

```

```
#densite spectrale du bruit
sigma_b=var_b*tx*ty
#transformee de Fourier du noyau
k=fft2(KK)
#fonction de multiplication
mul=np.conj(k)/(abs(k)**2+(sigma_b/sigma_s))
#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
```