Nicolas Billy

TP 2 IMA

2 Transformation géométrique

Figure 1: Lena rotation interpolation plus proche voisin

Figure 2: Lena rotation interpolation bilinéaire

On effectue maintenant 8 rotations de 45 degrés, pour voir la différence avec l'image initiale on regarde la valeur absolue de la différence avec l'image originale:

Figure 3: Lena 8 rotation interpolation plus proche voisin

Figure 4: Lena 8 rotations interpolation bilinéaire

L'image de droite possède plus de zones noires, cela veut dire qu'elle est plus proche de l'image originale. En mettant le facteur d'homothétie plus petit que 1, l'image apprait dézoomée. De plsu en zoomant, on observe que l'image est plus pixelisée et que des rayures apparaissent, signes d'un repliement spectral. Il faut donc appliquer un filtre passe bas avant d'effectuer la manipulation pour éviter ces problèmes.

3 Filtrage linéaire et médian

- Le paramètre de la commande *get_kau_ker* s représente l'écart type de la gaussienne générée, en fonction de cet écart type on aura une fenêtre plus ou moins grande pour stocker la gaussienne.

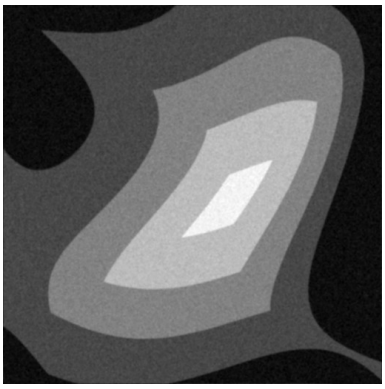

Figure 5: filtre gaussien, 1

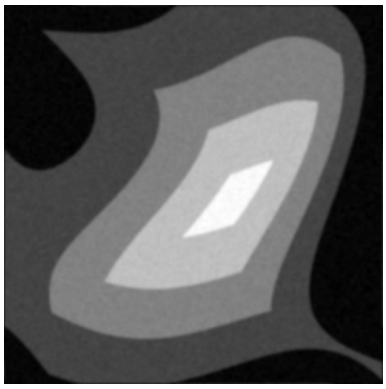

Figure 6: filtre gaussien, 2

Figure 7: filtre gaussien, 5

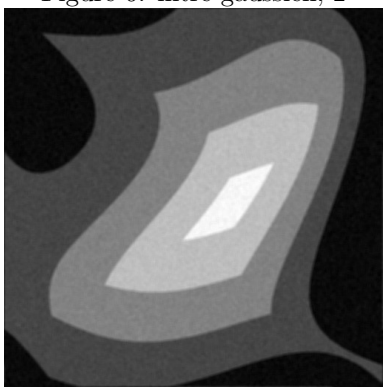

Figure 8: filtre constant, 5

- Pour évaluer le bruit, on peut calculer la variance de l'image dans une zone que l'on sait homogène, comme dans un carré en bas à gauche pour l'image carre_orig.tif.
- comparaison filtrage linéaire/médian sur une image bruitée. Pour cela on visualise la différence de la nouvelle image filtrée avec l'orginale pour comparer les filtrages.

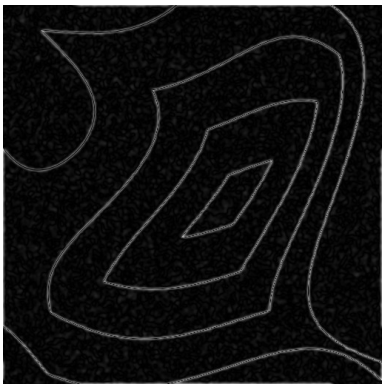

Figure 9: Filtrage gaussien, ecart type 2

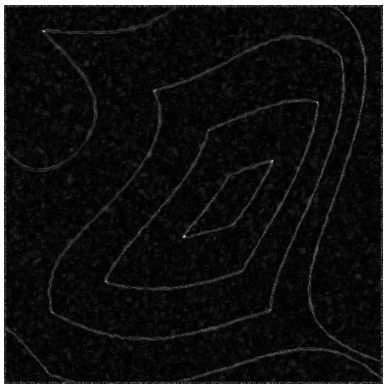

Figure 10: filtre médian, r=3

- On utilise maintenant l'image pyramide_pulse.tif, on constate que le filtre médian est plus précis pour le rendu des bords, sauf au niveau des coins, ce filtre les arrondit.

Figure 11: Filtrage gaussien, ecart type 2

Figure 12: filtre médian, r=3

- Comparaison linéaire/médian (figure 13). On peut visualiser des niveaux de bruits sur une ligne de l'image, on voit qu'avec le filtre médian, il n'y a presque plus de bruit résiduel sur les plateaux. D'ou l'aspect général moins flou de la figure 12.

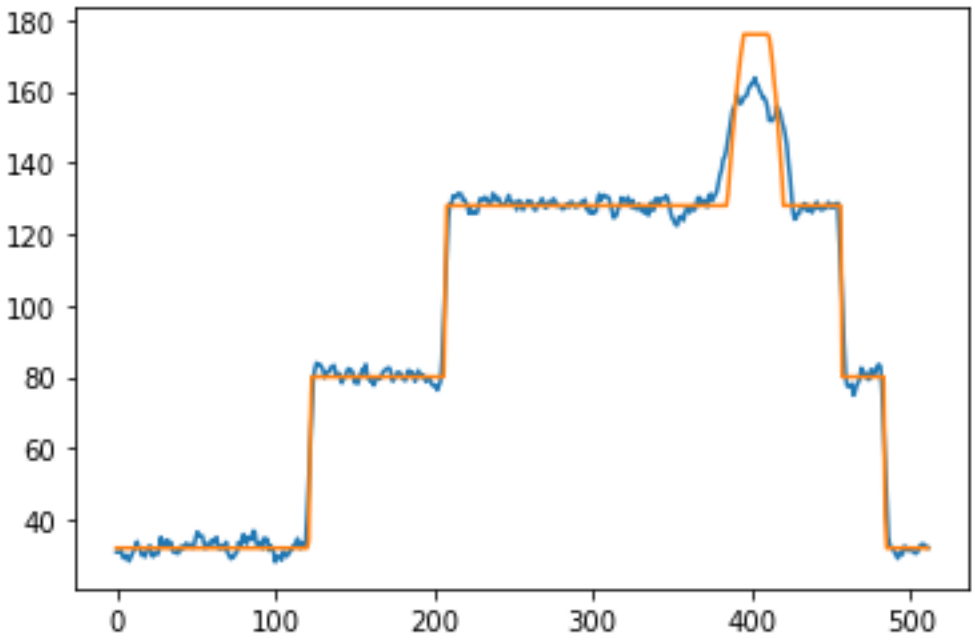

Figure 13: Visualisaiton des niveaux de gris sur une ligne de l'image. (Bleu: filtre gaussien, jaune: filtre médian)

- Avec le filtrage médian, le petit point lumineux disparait, car en faisant la moyenne avec les autres carrés noirs, le point est insignifiant, alors qu'avec le filtrage linéaire avec un noyau gaussien on observe un léger point flou.

4 Restauration

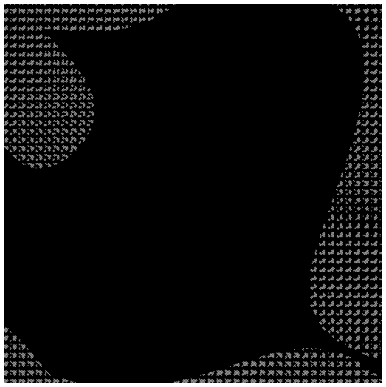

Figure 14: Valeur absolue de la différence entre l'inversion de la photo sans bruit et la photo originelle,pou la photo pyramide

- On voit qu'il y a l'apparition de bruits dnas les zones qui étaient noires. En rajoutant du bruit après le filtrage, on ne reconnait plus du tout l'image restaurée par filtre_inverse.

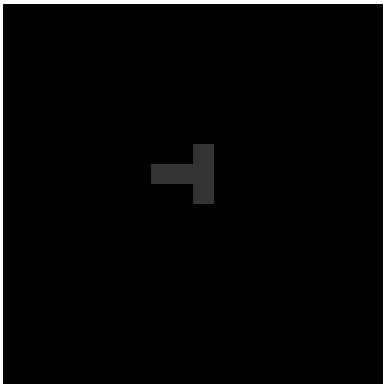

Figure 15: Zoom sur l'image carre_flou.tif dans la zone ou le carré devrait apparaitre

- On zoom dans la zone ou le point lumineux était (figure 15), il a été transformé en une nouvelle figure: on voit 9 pixels blancs, le noyau de convolution peut donc être: $K = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$

- Différentes images de la restitution du carré pour différentes valeurs de λ dans le filtre de Weiner en utilisant le noyau de convolution K, après avoir rajouté du bruit à l'image originale. Il semble que la restituion se fasse mieux pour $\lambda = 5$

Figure 16: $\lambda = 0.1$

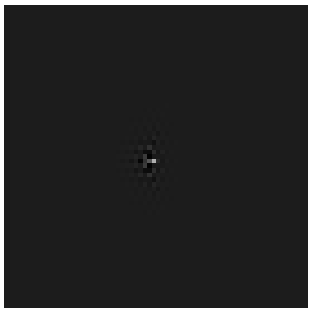

Figure 17: $\lambda = 1$

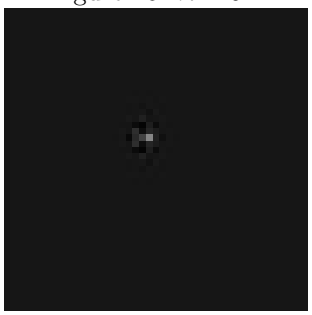

Figure 18: $\lambda = 5$

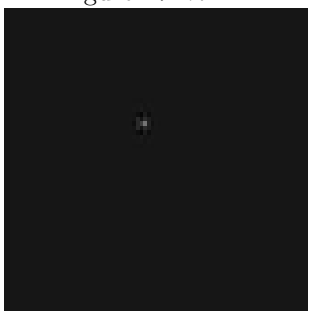

Figure 19: $\lambda = 10$

5 Applications

Comparaison filtrage linéaire et médian

Pour trouver la taille du noyau constant qui réduit le bruit dans les mêmes proportions qu'un filtre médian, on peut tester différentes valeurs de taille de fitlre constant en partant de 0. Pour comparer les niveaux de bruit, on regard la variance de l'image dans un patch ou l'on sait que les niveaux de gris sont homogènes, par exemple dans un rectangle en bas de l'image. On stock les variances calculés pour chaque image dans une liste, et on regarde ensuite quel élément minimise la différence de la valeur absolue avec la variance de l'iamge calculée dans le même patch pour un filtre médian.
On trouve ainsi l'indicce qui correspond à la taille du noya constant: $t = 3$ avec une variance du patch: $var\_patch = 3.3828340979841216$ et la variance de l'image filtrée avec le filtre médian: $var\_filtre\_med = 3.204868505028925$
Code qui implémente cet algorithme:

```
im = skio.imread('images/carre_orig.tif')
imbr = noise(im,5)
im_filtre_med = median_filter(imbr,typ=2 ,r=4)
x0=0
y0=0
x1=256
y1= 50
#Patch rectangulaire en bas de l'image, dans la partie sombre
variance_disk = var_image(im_filtre_med, x0, y0, x1, y1)
List_var=[]
for i in range(1,15):
    Filtre_constant = get_cst_ker(i)
    imbr_filtr = filtre_lineaire(imbr, Filtre_constant)
    List_var.append(var_image(imbr_filtr, x0, y0, x1, y1))
    #On stock les valeurs de variance trouvees pour ce patch avec le filtrage constant

min_value = np.abs(List_var[0] -variance_disk)
for e in range(0, len(List_var)-1):
    Dif = np.abs(List_var[e] -variance_disk)
    if Dif < min_value:
        min_value = Dif
        min_index = e
```

Calcul théorique paramètre de restauration

```
    def wiener_modified(im,K):
    """ inversion en utilisant le spectre de l'image degrade  a la place de lambda omega2
       """
    fft2=np.fft.fft2
    ifft2=np.fft.ifft2
    (ty,tx)=im.shape
   [.... meme code...]
    #tranformee de Fourier de l'image degradeee
    g=fft2(im)
    #transformee de Fourier du noyau
    k=fft2(KK)
    sigma_b = np.abs(g**2)
    sigma_s = var_image(im, 0, 0, 256, 256)*tx*ty
    factor = sigma_b**2/(sigma_s**2)
    mul=np.conj(k)/(abs(k)**2+factor
    ....suite meme code
```