Télécom PARIS

Compte Rendu :
Travaux pratiques – IMA 201
Filtrage et Restauration

Hiba Dahmani

Etudiante en $2^{éme}$ année

Filière IMA

Année universitaire : 2021/2022

2. Transformations géométriques

i)
les transformations géométriques reviennent à un problème d'interpolation c'est-à-dire on cherche à calculer la valeur de l'image dans des coordonnées ou on n'a pas la valeur précisément .
La différence dans les méthodes à plus proche voisin et bilinéaire est dans les régularités qu'on suppose concernant l'image
Dans le cas du plus proche voisin on suppose que l'image est constante pas morceaux (des pavés en 2D) or dans le cas d'interpolation bilinéaire on suppose que l'image est linéaire par morceaux suivant les 2 dimensions donc elle prend en compte les 4 voisins les plus proches.

Rotation de 45° par la méthode du PPV

Rotation de 45° par la méthode bilinéaire

L'interpolation bilinéaire a permis d'avoir une image moins pixélisé et plus lisse .
ii)

8 rotation de 45° avec interpolation bilinéaire

8 rotations de 45 avec interpolation ppv

l'effet de l'interpolation ppv est plus clair après 8 rotations ,L'image est très pixélisée.
iii)
Si on applique la rotation avec un facteur de zoom 1/2 , on remarque l'apparition du phénomène de l'aliasing ceci est très clair au niveau des cheveux .
Pour atténuer ce phénomène peut être atténué en appliquant un filtre passe-bas avant de procéder le sous échantillonnage . Il est aussi possible d'appliquer un filtre moyenneur ou médian après la transformation.

3. Filtre linéaire et médian

I)
Le rapport entre s standard déviation et taille du noyau renvoyé ss par get_gau_ker est donné par la formule suivante :
Ss=int(max(3,   2*np.round(2.5*s)+1)

ii)

pyramide,tif

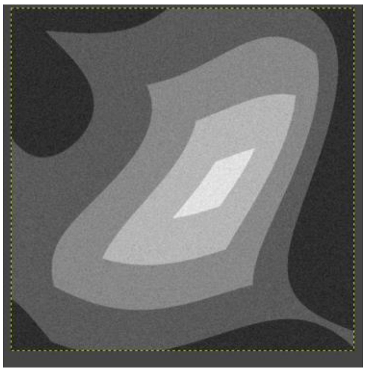


image bruitée

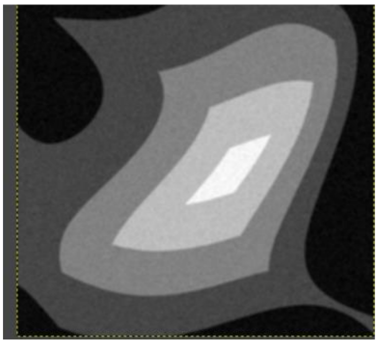


Image après application du filtre constant

Cas du filtre gaussien

Évaluation de l'effet des filtres par calcul de la variance du bruit résiduel

```
In [51]: var_image(imbr,105,8,132,26)
Out[51]: 108.83409719205422

In [52]: var_image(fc,105,8,132,26)
Out[52]: 3.7242771490268156

In [53]: var_image(fg,105,8,132,26)
Out[53]: 0.3367376037573473

In [54]:
```

fc : filtre constante , fg : filtre gaussien.
Pour estimer le bruit présent dans une image, il est difficile de l'évaluer en calculant la variance sur l'image entière car généralement l'image par nature contient plusieurs zones de natures différentes (contour , couleur , textures) donc il est plus logique de calculer la

variance dans une zone homogène . Après application des filtres la variance dans une zone homogène de l'image diminue. Le bruit est très réduit surtout après l'application du filtre gaussien.

iii)

Application du filtre médian

```
In [57]:

In [57]: fm=median_filter(imbr,r=6)

In [58]: var_image(fm,105,8,132,26)
Out[58]: 0.6549072131561227
```

iiii) Comparaison linéaire/médian sur pyra-impulse.tif

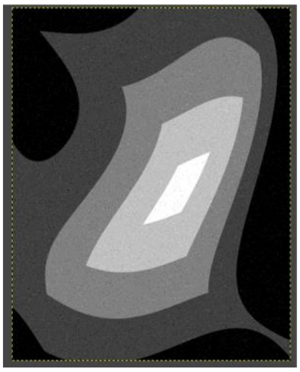
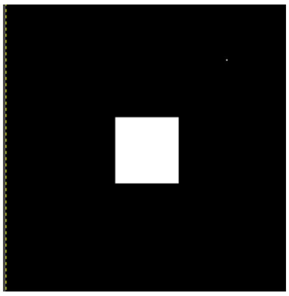


Image originale

Image filtrée : filtre médian   filtre linéaire constant   filtre gaussien linéaire

Par principe , le filtre médian cherche la valeur médiane dans un patch de pixels ce qui fait que ce filtre ignore totalement les valeurs extrêmes tel que le bruit présent dans l'image (les taches) .
Or les filtres linéaires donnent une valeur suite à des combinaisons de toutes les valeurs des pixels dans le patch . Le filtre gaussien réussit à éliminer le bruit mais produit un flou.



Carré_orig.tif

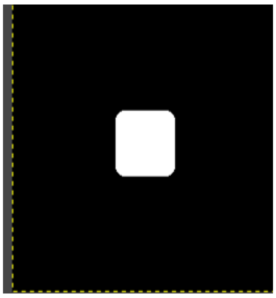


Image filtrée par le filtre médian

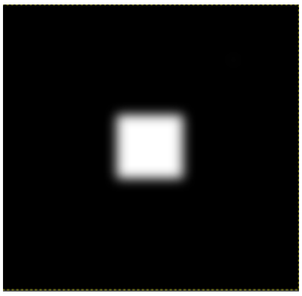


Image filtrée par un filtre linéaire gaussien

Le filtre médian élimine totalement le point lumineux en haut droite de l'image originale car dans le voisinage de ce point , tous les autres points sont noirs ce qui fait que ce point correspond à un point extrême qui est négligé Or dans le cas d'un filtre linéaire moyenneur cette valeur apparaît.

4. Restauration

i)

Image originale

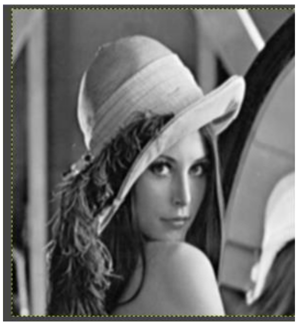


Image filtrée

Image restauré

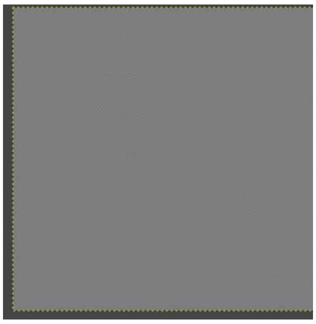
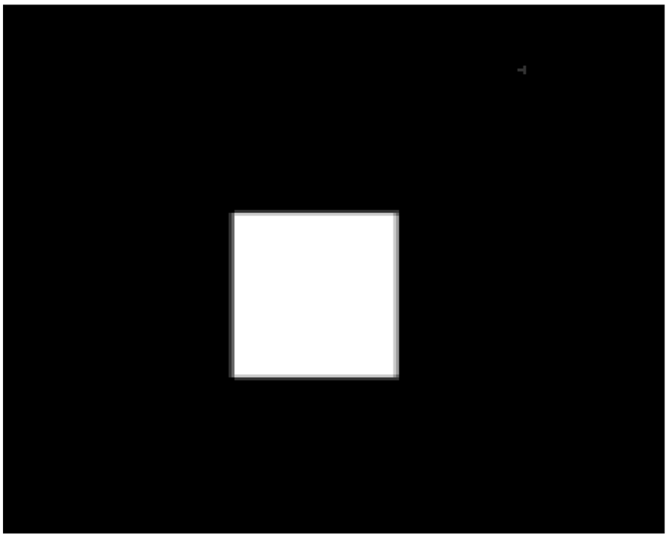


Image restauré après ajout de bruit

Le filtrage inverse a permis de retrouver identiquement l'image restaurée , mais l'ajout du bruit à l'image filtré a causé la perte totale de l'information

ii)

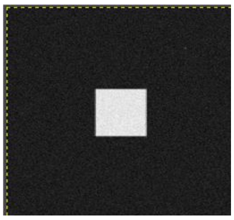
le noyau de convolution est la réponse impulsionnelle du filtre c'est-à-dire la réponse à un point lumineux.

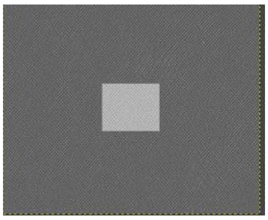


En zoomant sur l'image , on observe la réponse du filtre au point lumineux qui permet de déduire que le noyau du filtre appliqué à l'image est :

0	0	1/5
1/5	1/5	1/5
0	0	1/5

iii) Filtrage de Wiener





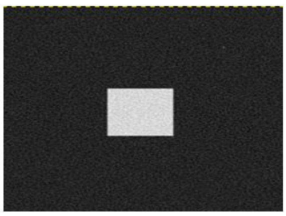


image bruitée filtrée par un filtre de Weiner avec λ=0 λ=6

Lors du la variation de ⋋ , on observe qu'il y a deux erreurs possibles soit une sous-estimation ou une sur estimation qui dans les deux cas ne permettent pas d'avoir une bonne restauration de l'image .

5. Application

5.1 Comparaison filtrage linéaire et médian

```
im=skio.imread('images/carre_orig.tif')
imbr=noise(im,5)
imfl=median_filter(imbr,r=4,typ=2)
var=var_image(imfl,0,0,256,256)
c=0
val=1e10
for i in range(3,13,2):
    mask=get_cst_ker(i)
    imm=filtre_lineaire(imbr,mask)
    varm=var_image(imm,0,0,256,256)
    if(abs(varm-var)<val):
        val=abs(varm-var)
        c=i
print(c)
print(val)
```

Cet algorithme permet de varier la taille du noyau du filtre constant jusqu'à trouver une taille qui permet de réduire le bruit dans les mêmes proportions qu'un filtre médian circulaire de rayon 4. On compare à chaque itération les variances des deux images filtrés .

```
   ...:     mask=get_cst_ker(i)
   ...:     imm=filtre_lineaire(imbr,mask)
   ...:     varm=var_image(imm,0,0,256,256)
   ...:     if(abs(varm-var)<val):
   ...:         val=abs(varm-var)
   ...:         c=i
   ...:
   ...:
   ...: print(c)
   ...: print(val)
3
73.07384442947523

In [111]:
```

La taille du filtre trouvé est 3.

5.2 Calcul théorique du paramètre de restauration :

On change la valeur de sigma par le carré de la TF du signal dégradé

```python
def wiener(im,K,sigma,sigbr):
    """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 degradeee
    g=fft2(im)
    #transformee de Fourier du noyau
    k=fft2(KK)
    #fonction de mutilplication
    mul=np.conj(k)/(abs(k)**2+sigbr/sigma)
    #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

im=skio.imread('images/carre_flou.tif')

K=np.array([[0,0,1/5],[1/5,1/5,1/5],[0,0,1/5]])
sig=np.abs(np.fft.fft2(im))**2
imbr=noise(im,10)
viewimage(wiener(imbr,K,sigma=sig,sigbr=10))
```

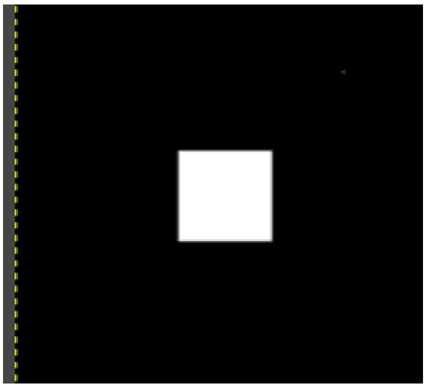


Image originale

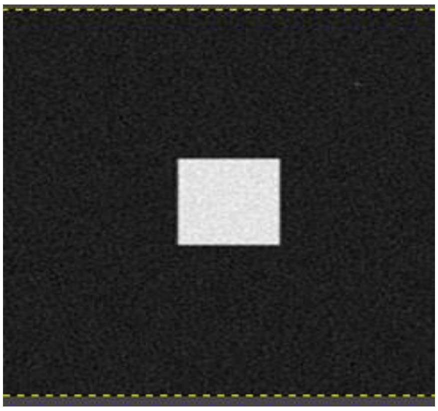


Image restaurée par l'ancienne fonction de Wiener

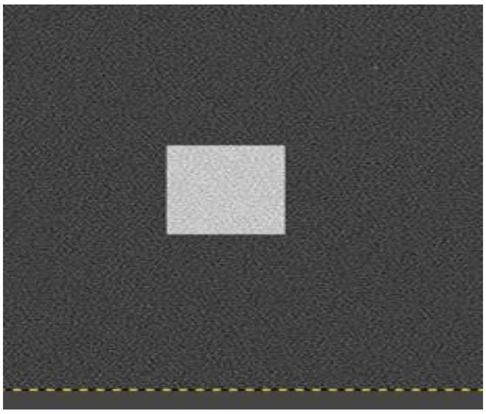


Image restaurée par la nouvelle fonction de Wiener