Introduction au Traitement des Images (IMA201)

TP2

Éleve: Vitor de Sousa França

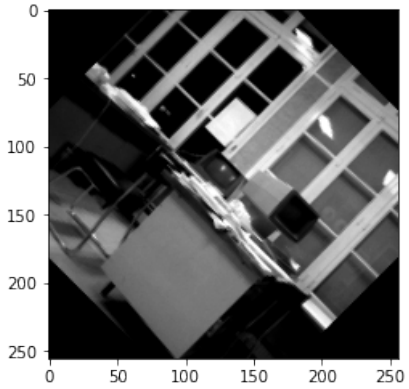
Email: vitor.franca@telecom-paris.fr

Septembre
2022

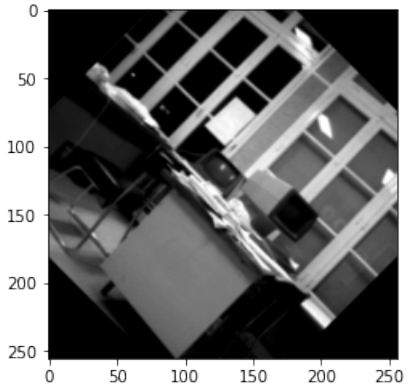
1 Transformation géométrique

1.1 Utiliser la fonction (rotation) pour transformer une image de votre choix. Quelle différence y-a-t-il entre la méthode à plus proche voisin et la méthode bilinéaire ?

On peut voir sur les Images 1(a) et 1(b) que lorsque on fait une simple rotation de 45 degrés sur l'image, il n'y a pas de différences visibles entre les résultat en utilisent des méthodes plus proche voisin (ppv) et bilinéaire.



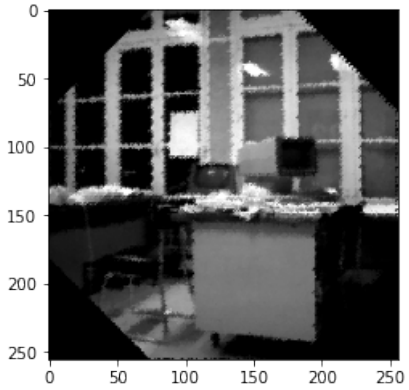
((a)) rotation en utilisent ppv



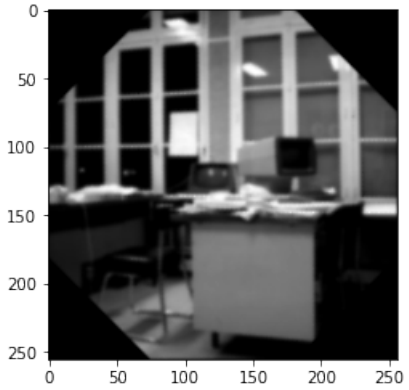
((b)) rotation en utilisent bilinéaire

1.2 Que constatez-vous sur une image qui aurait subi huit rotations de 45 degrés (en bilinéaire et en plus proche voisin)?

Toutefois, comme on peut voir sur les Images 2(a) et 2(b) lorsque l'opération de rotation est répétée 8 fois, les différences entre les méthodes deviennent visibles. Tandis que dans la méthode bilinéaire l'image devient plus floue et dans la méthode plus proche voisin certains pixels des bords perdent leur information d'origine en prenant la valeur d'un pixel proche d'un niveau de gris complètement différent.



((a)) 8 rotations en utilisent ppv



((b)) 8 rotations en utilisent bilinéaire

1.3 Que constatez-vous si vous appliquez la rotation avec un facteur de zoom inférieur à 1 (par exemple 1/2) ? Qu'aurait-il fallu faire pour atténuer l'effet constaté ?

Quand on utilise un facteur de zoom inférieur à 1 l'image reduire le quantitée des pixels et pourtent l'information. Ce que on voit, c'est l'effet d'aliasing et pour le reduire on peut utiliser un filtre passe-bas avant de procéder .

2 Filtrage linéaire et médian

2.1 Expliquer le rapport entre la taille du noyau (size) renvoyé par get_gau_ker et le paramètre cette commande.

Le paramètre "s" determine combien des pixels et leurs poids respectifs pour être utilisé dans l'opération de convolution. La taille du noyau "(SSxSS)" est determiné par l'equation 1.

$$\begin{aligned} SS &= 3 && \text{, si } s < 1 \\ SS &= 2 \cdot round(2.5 \cdot s) + 1 && \text{, si } s \geq 1 \end{aligned} \tag{1}$$

Le résultat de cette équation est toujours impair avec la plus petite taille étant une matrice 3x3. L'augmentation de 0.5 en s implique une augmentation de 2 en "SS" (le prochaine l'impair), c'est-à-dire l'ajout de 4 pixels à la convolution.

2.2 Après avoir ajouté du bruit à une image simple telle que pyramide.tif ou carre_orig.tif et avoir filtré le résultat avec des filtres linéaires, expliquez comment on peut evaluer (sur des images aussi simples) la quantité de bruit résiduel (la commande var image donne la variance d'une partie d'une image).

On peut evaluer la quantité de bruit résiduel à travers une région uniforme de l'image. Donc sur l'image carre_orig.tif on calcule le variance en utilisent une région qui ne change pas de couleur.

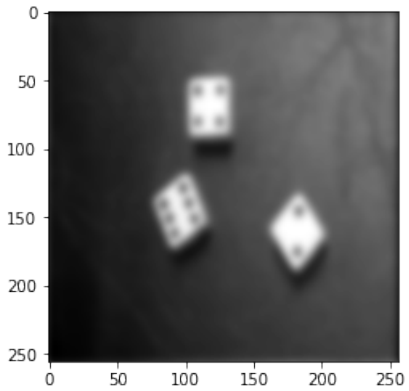
Table 1: Bruit Résiduel

Image	Variance
Sans bruit	0
Bruitée	23.34
Filtrée	0.405

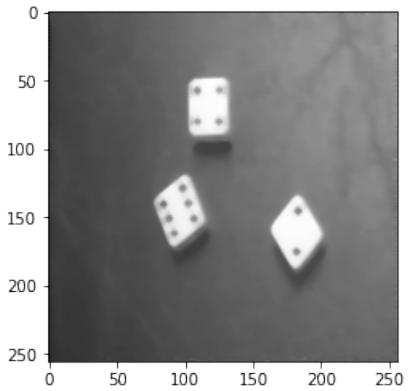
On peut voir sur le Table 1 que l'image carre_orig.tif filtrée avec un filtre linéaire après l'adition du bruit blanc gaussier d'écart-type égal à 5 a une bruit résiduel de variance 0.405.

2.3 Appliquer un filtrage médian à une image bruitée et comparer le résultat avec un filtrage linéaire.

On peut voir sur les Images 3(a) et 3(b) que en utilisent un filtre médian l'image est moins floue que quand l'image subit un filtrage linéaire. Toutefois il y a aussi moins de niveuax de gris.



((a)) l'image resultent d'utilisation du filtre linéaire

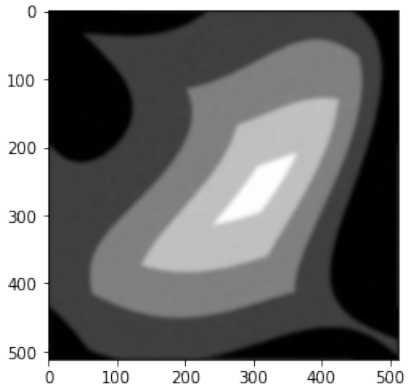


((b)) l'image resultent d'utilisation du filtre médian

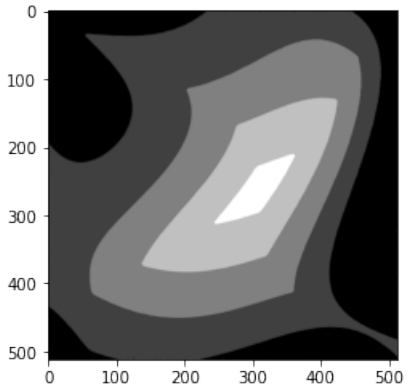
2.4 Faites une comparaison linéaire/médian sur l'image pyra-impulse.tif. Que constatez-vous ? Expliquer la différence de comportement entre filtrage linéaire et médian sur le point lumineux situé en haut à droite de l'image carre orig.tif

.

En comparant les Images 4(a) et 4(b) en peut voir que le filtre median a supprimé du bruit sans distorsions visibles dans l'image. Par contre, le filtre linéaire rendu l'image floue.

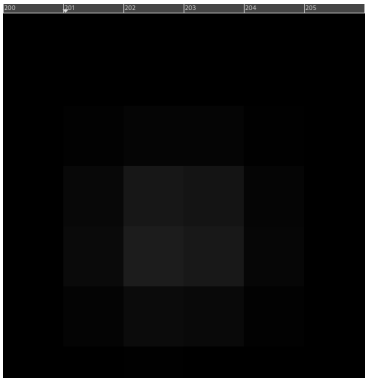


((a)) l'image resultent d'utilisation du filtre linéaire

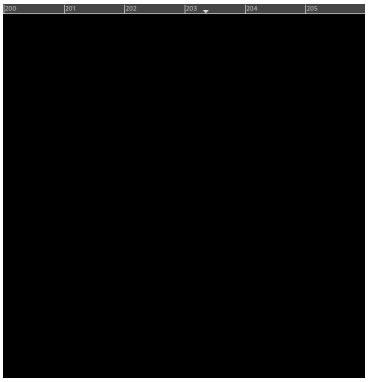


((b)) l'image resultent d'utilisation du filtre médian

En plus, on peut voir que sur l'image carre_orig.tif, quand on utilise le filtre linéaire, l'image 5(a), le carré devenu moins gris (plus proche du noire) en raison de la valeur grise moyenne de la région et aussi plus étendu (avec une plus grande surface). En revanche, quand on utilise le filtre median le carré disparaît complètement, l'image 5(b).



((a)) l'image resultent d'utilisation du filtre linéaire



((b)) l'image resultent d'utilisation du filtre médian

3 Restauration

3.1 Appliquer un filtre linéaire à une image puis utilisez la fonction filtre_inverse. Que constatez-vous? Que se passe-t-il si vous ajoutez très peu de bruit à l'image floutée avant de la restaurer par la commande précédente ?

Dans le premier cas quand on utilise le filtre linéaire l'image devenu un peu floue et après l'utilisation de la fonction filtre_inverse l'image résultante ce, visuellement, inchangée en comparaison l'original.

Par contre, quand on ajute un peu de bruit à l'image resultant du filtre linéaire (floutée) et utilise le commande précédente le résultat est un image avec seulement du bruit.

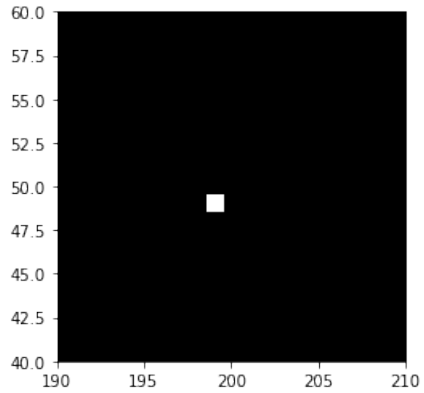
3.2 Comment pouvez-vous déterminer le noyau de convolution qu'a subi l'image carre_flou.tif

Il y a un pixel sur le haut droit de l'image carre_flou, image 6(a), ce que on peut comprendre comme une delta de Dirac. Quand on utilise une filtre, ce que on fait est de convoluer l'image avec le filtre. En plus, quand on fait une convolution oú l'image est une delta de Dirac le résultat est le filtre lui-même. On peut voir sur l'image 6(b) que le kernel est donc.

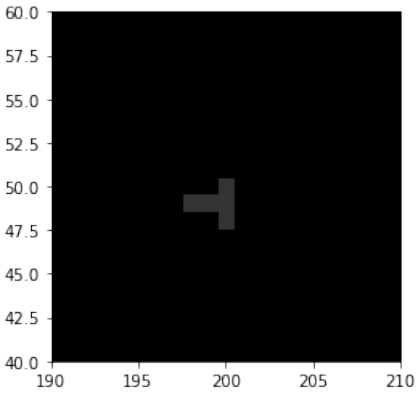
$$K = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

3.3 Après avoir ajouté du bruit à cette image utilisez la fonction wiener pour restaurer cette image. Faites varier le parametre λ et commentez les résultats.

Quand on augmente le parametre λ le filtre reduire de plus en plus le bruit jusqu'au moment où cela rend également l'image floue (surestimation du bruit). Pour un cette image avec un bruit d'écart-type 0.5, $\lambda = 250$ a seulement un peu de bruit et un peu de floue.



((a)) l'image originale



((b)) l'image après le filtre

4 Applications

4.1 Comparaison filtrage linéaire et médian

4.1.1 Pour une image simple telle que carre orig.tif et un bruit d'écart-type 5, trouver la taille du noyau constant qui réduit le bruit dans les mêmes proportions qu'un filtre médian circulaire de rayon 4. (explicitez l'algorithme utilisé)

```
im = skio.imread(path+'carre_orig.tif') #load image
im_noise = noise(im, 5) #add noise

im_med = median_filter(im_noise, typ=2, r=4) #using median filter
var_im_med = var_image(im_med,x0=0,y0=0,x1=50,y1=50) #getting image
    variance from an uniform part

diff = float('inf') #initializing variance
size = 0

#we increase the size until the var_im_med be (approx) equal to var_im_lin
while diff > 0:
  size += 1
  im_lin = filtre_lineaire(im_noise, get_cst_ker(size))
  var_im_lin = var_image(im_lin,x0=0,y0=0,x1=50,y1=50)
  diff = var_im_lin - var_im_med

print(size)
```

La taille du noyau constant qui réduit le bruit dans les mêmes proportions qu'un filtre médian circulaire de rayon 4 etait 3x3 .

4.2 Calcul théorique du paramètre de restauration

4.2.1 Ici on travaille à nouveau sur l'image carre flou.tif que l'on bruite et restaure par wiener. Modifiez la fonction wiener afin qu'elle utilise le spectre de l'image dégradée à place de $\lambda\omega^2$.

```
def wiener(im,K,lamb=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 degradeee
    g=fft2(im)
    #transformee de Fourier du noyau
    k=fft2(KK)

    #modification
    var_signal = g**2
    var_bruit = var_image(im,x0=0,y0=0,x1=50,y1=50)

    #fonction de mutilplication
    mul=np.conj(k)/(abs(k)**2+ var_signal/var_bruit )
    #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
```