Giovanni Benedetti da Rosa

TP2- IMA 201

2 Transformation géométrique
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 linéaire ?

Quand on fait une rotation dans une image, le résultat de pixels ne correspond pas au lieu des pixels originales, donc il faut faire une interpolation. Ainsi, dans la méthode de plus proche voisin, ça consiste à sélectionner le pixel le plus proche de la position de destination lors de la rotation. La méthode bilinéaire effectue une interpolation entre les pixels voisins en utilisant une fonction linéaire. Ainsi, en général la méthode de plus proche voisin produit une image moins lisse par rapport le bilinéaire.

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

L'image de plus proche voisin est plus lisse, avec une qualité pire. Ça a certainement une relation avec la méthode d' interpolation utilisée. Le méthode de plus proche voisin fait a chaque rotation une approximation grossière qui s' accumule, tanids que dans la méthode bilineaire un flous se accumulle.

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 ?

Un pixel sur la nouvelle image représente une surface plus grande qu'un pixel sur l'ancienne image, et donc utiliser une seule valeur pour cette approximation peut être une mauvaise approximation(conforme le théorème de shannon d'échantillonnage). Ainsi, on peut utiliser un filtre passe-bas, avant de réduire la taille.

Si T est la transformation dézoom, g une filtre passe-bas I l'image originalle et J l'image après la transformation on a que:

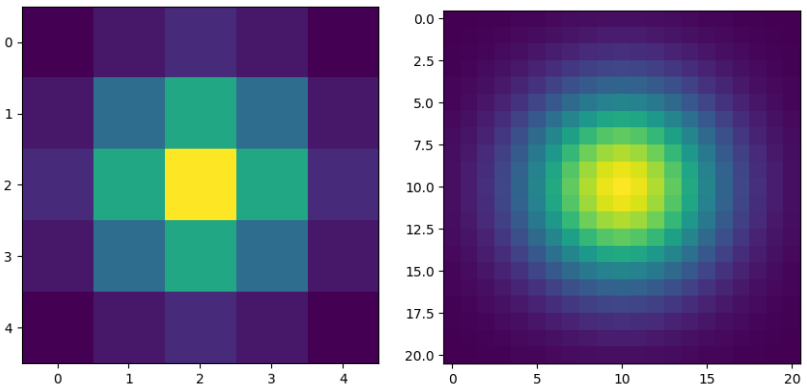
$$\tilde{I} = I * G \Rightarrow J(k,n) = \tilde{I}(T^{-1}(k,n))$$

Lena dézoomée

3 Filtrage linéaire et médian

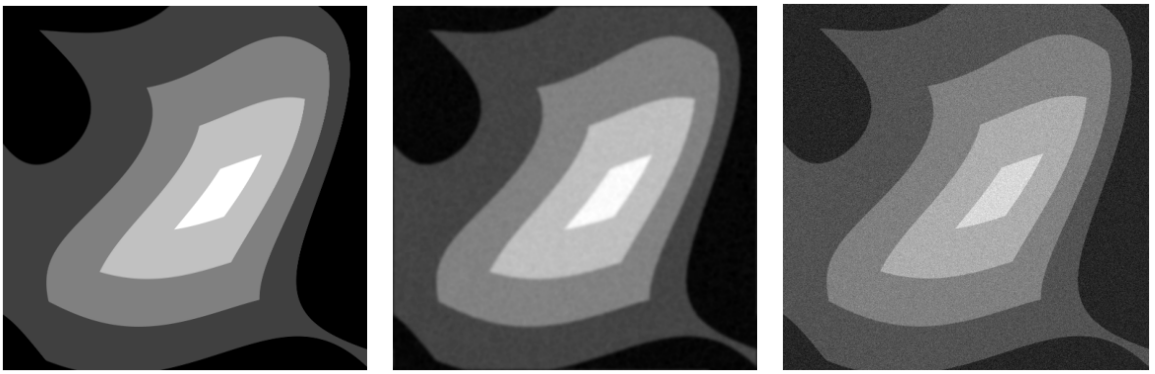
Filtrage linéaire et médian

Expliquer le rapport entre la taille du noyau (size) renvoyé par get gau ker et le paramètre cette commande.



Filtre de écart-type 1 et 4

Le taille du noyau dépend de le écarte et est representé par une matrice carré, pour créer un filtre gaussiene de ce taille

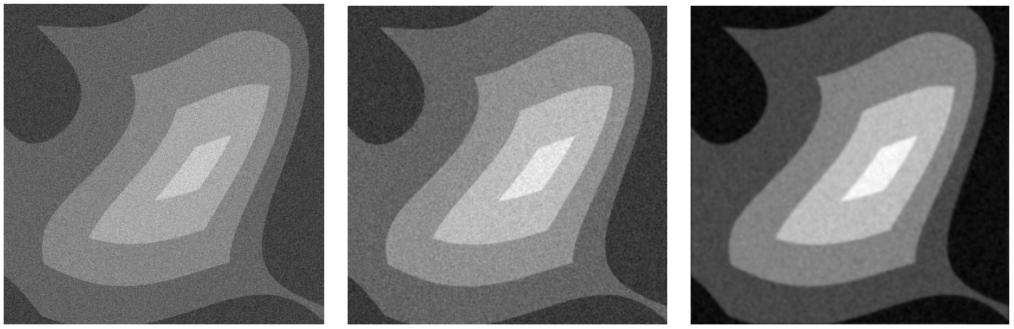


L'image original avec noyau et après le filtrage linéaire

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.

Nous pouvons mesure la variation de niveaux des gris dans une certaine région apparemment uniforme, encomparant cette variation avant et après application du filtre. C'est ce que fait la fonction var_image, à travers la variance.

Appliquer un filtrage médian à une image bruitée et comparer le résultat avec un filtrage linéaire. Faites une comparaison linéaire/médian sur l'image pyra-impulse.tif.



Comparation entre l'image avec bruit, avec un filtre lineaire et avec un filtre median

- variance filtremédian 2.67
- variance filtre linéaire 152.18

Dans ce cas on peut constater que la variance du filtre linéaire a été plus grand qui dans le filtre linéaire. Comme j'ai ajouté beaucoup de bruit(sd = 10), je pense qui le filtre médiane a meillure performé.

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.

original, linéaire, médian

Après le filtrage linéairere, reste une zone plus blanche dans le voisin proches a le point blanc, une fois que toute les valeurs sont utilisées dans le calculus.

Pourtant, dans le filtre median, le point blanc est un *outlier,* un point qui a completament disparu après la convulotion.

4-Restauration

Appliquer un filtre lin'eaire à une image puis utiliser la fonction filtre inverse 2 . 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 ?

Utilisation du filtre inverse

Aprés ajouter de bruit

Les images naturelles elles-mêmes contiennent beaucoup de basses fréquence et très peu de haute frequènces. Ainsi, comme le bruit est equale dans toutes les frequènces, après la

tranformée inverse qui multiplie beaucoup les composants de hautes fréquences. Ceci explique la différence entre la transformée inverse avec et sans bruit.

Comment pouvez-vous déterminer le noyau de convolution qu'a subi l'image carre flou tif ? Dans l'image originalle il y a un point blanc le dans l'image aprés da ajouter de bruit, ça fonctionne comme un delta de dirac et filtre le signal de convolution en le faisant passer toute le signaux qui est convolué $I*h = h$. De cette manière, en comparant la forme du point blanc dans l'image originale et dans l'image bruitée, nous constatons que la matrice qui a été convoluée doit avoir une forme similaire à $\begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$.

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

Comparaison de restauration avec $\lambda = 15$ et $\lambda = 60$

On peut constater que tandis que le valeur de lambda augmente jusqu'à une limite. Après ça, l'image reste floue, parce que dans ce cas le bruit a été sous-estimé.

5 Application

Pour une image simple telle que carré original 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.

```
im = noise(im, 5) # bruit dans l'image
long = 100 #création de variable pour le loop
median_var = var_image(median_filter(im, typ=2, r=4), 0,0,50,50) #calcule de la variance de le median filter
#loop pour faire varier le noyau
for i in range(1,50):
    linear_var = var_image(filtre_lineaire(im, get_cst_ker(i)), 0,0,50,50)#variance de filltre lineaire avec  noyau cte
    if (abs(linear_var-median_var)/median_var) < long:# verifie la diffèrence entre les filtres normalisé
        long  = abs(linear_var-median_var)/median_var #prendre le valuer plus grand
```

index = i # prendre le index

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 $\lambda\omega^2$.

La valeur de $\sigma_s^2(\omega)$ sera prise comme ágale au carré de la TF du signal dégradé et la valeur σ_b^2 sera la variance du bruit multiplié par le nombre de pixels de l'image pour une raison de normalisation de la TF.

```
def wiener_modifie(im,K,lamb=0):
    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)

    sigma_s = (g)**2
    sigma_b = im.var()*len(im)/sigma_s
    #fonction de mutilplication
    mul=np.conj(k)/(abs(k)**2+(sigma_s/sigma_b)*w2)
    #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
```