Travaux pratiques 2 - IMA201- Réponses

Júlia Togashi de Miranda

2. Transformation Géométrique

L'image Lena originale et avec un rotacion de 45 degrees, utilisant respectivemente la methode plus proche de voisin et bilineire

Quelle diference y-a-t-il entre la méthode à plus proche voisin et la méthode bilinéaire ?

C'est le methode de interpolation que on utilise pour tourner l'image. Quand on fair la rotation, les cordenades que on trouve n'est sont pas entiers, donc il faut que on utilise la interpolation.

Dans le methode de plus proche voisin chaque pixel est créé par « l'agrandissement »du pixel le plus proche de lui (voisin). Donc, pour agrandir une image avec un facteur n, on prendre un bloc de nxn pixels identiques. Dans la méthode bi-linéaire s'obtient par interpolation linéaire dans chaque dimension successivement. De cette manier, on peut trouver un meiller resultad avec le métode bilinéaire (à la photo le nez est plus clair dans la deuxieme, par example).

L'image Lena subi huit rotacion de 45 degrees, utilisant respectivemente la methode plus proche de voisin et bilineire

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

Celui qui utilise la méthode du plus proche voisin aggrave beaucoup la qualité, car il passe par chaque rotation en faisant des interpolations grossières qui entraînent une perte de qualité d'image, ce qui n'est pas si évident dans la méthode bilinéaire.

Rotation avec un facteur de zoom=1/2 appliqué sur l'image Lena

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 ?

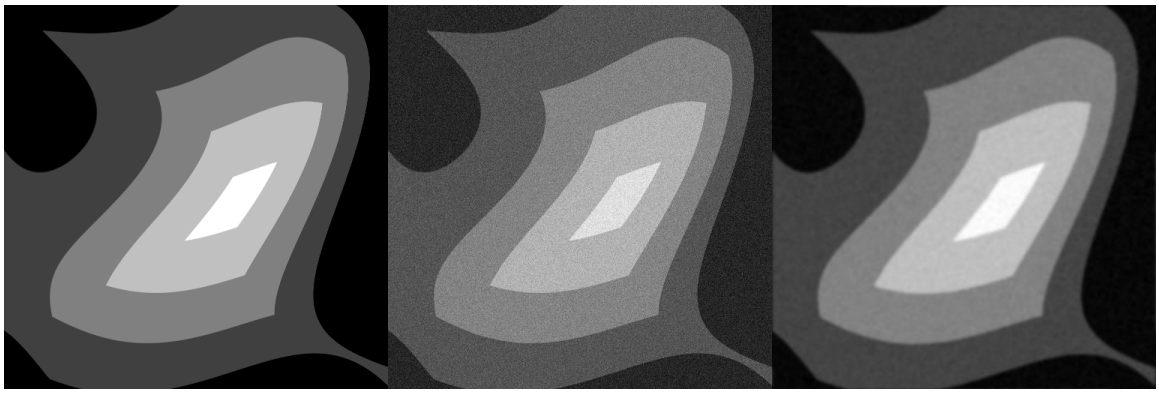
A chaque fois que on fait la rotation de l'image, elle est réduite pour le facteur du zoom. Donc avec une rotation l'image va avoir ½ de la taille. Deux rotations, ¼. Pour atténuer cet effet, on utilise l'option clip égale à faux.

3. Filtrage linéaire et médian

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

La taille du noyau renvoyé par get_gau_ker est un matrix carré de taille deux fois plus un du maximum entre l'arrondi du paramètre fois 2.5, fois deux plus un et le nombre trois.

La largeur du filtre est donnée par son écart-type σ, qui est le paramètre utilisé.

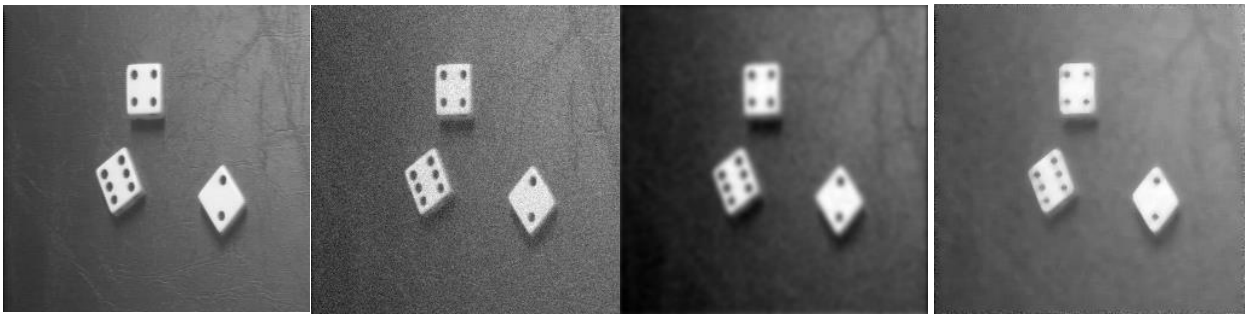


L'image pyramide original, quand on ajoute le bruit et après passer sur le filtre linéaire

Var_image (0,0,1,1) :
Original : 0.0
Bruit : 43.763389235844016
Filtre linéaire : 0.013374593538836573

Expliquez comment on peut évaluer (sur des images aussi simples) la quantité de bruit résiduel :

Nous avons que l'image initiale contenait des zones de couleur unie bien définies. Nous pouvons observer la quantité de bruit résiduel en regardant la variation de ton gris dans ces zones après le passage du filtre.



L'image dice1 original, quand on ajoute le bruit, après passer sur le filtre linéaire et après passer sur le filtre médien

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

Le filtre médian calculé pendre la médiane des voisins de chaque pixel. On peut voir, que le filtre médian a un effet de débruitage similaire dans la zone bruitée. L'avantage de faire le calcul de la médiane, c'est qu'elle n'est pas affectée pour des outliers comme le moyen. Donc il trouve très bons résultats pour supprimer le bruit impulsionnel, quand nous avons des pixels isolés de valeur très différente cote à cote.

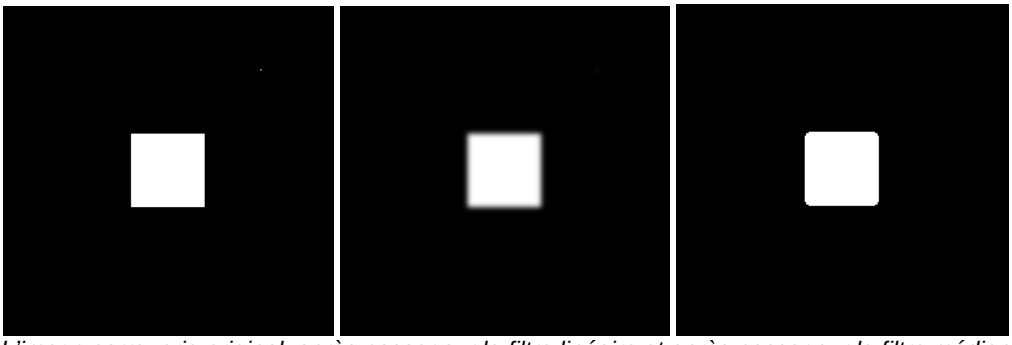
D'autre part, le filtre gaussien favorise le centre du voisinage, avec des coefficients plus grands au centre du masque (comme la distribution gaussienne bidimensionnelle). Il permet ainsi de réaliser un débruitage de l'image, avec une bonne conservation des contours, mais les contours sont moins nets que le filtre médian. Autre conséquence c'est que les zones sombres deviennent encore plus sombres.

On peut donc en conclure que le résultat du filtre médian ressemble plus à l'image d'origine qu'au filtre gaussien.

L'image pyra-impulse original, après passer sur le filtre linéaire et après passer sur le filtre médien

Faites une comparaison linéaire/médian sur l'image pyra-impulse.tif. Que constatez-vous ?

Réponse contenue dans la question ci-dessus.



L'image carre_orig original, après passer sur le filtre linéaire et après passer sur le filtre médien

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.

Nous avons un point blanc autour des points noirs. Pour le filtre linéaire, moyenneur ou gaussien, cela influencera les points voisins, parce que on l'utilise au calcul. Il sera atténué car ce n'est qu'un point, mais on peut encore voir une zone un peu plus claire dans le résultat final.

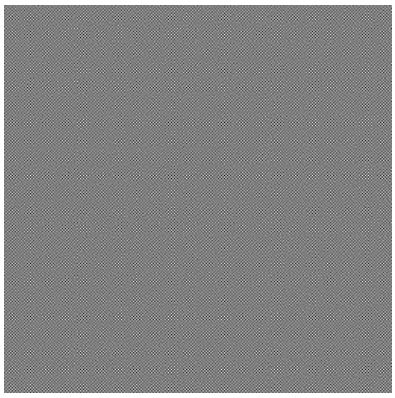
Quant au filtre médian, le point blanc est comme une valeur aberrante. Si nous avons plusieurs points noirs et un point blanc (0 0 0 0 0 0 0 0 0 1 1), la médiane restera un point noir (0), donc le point blanc disparaîtra complètement de l'image (cela a un effet curieux sur les bords du carré.)

4. Restauration

L'image carre_orig original, après passer sur le filtre linéaire et après passer sur le filtre inverse

Que constatez-vous ?

Après avoir appliqué la transformée inverse, nous récupérons l'image initiale



L'image quand on ajoute de bruit après appliquer le filtre inverse

Que se passe-t-il si vous ajoutez très peu de bruit à l'image floutée avant de la restaurer par la commande précédente ?

On obtient un bruit après avoir appliqué l'inverse

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

Nous pouvons déterminer le noyau regardant le point blanc qui a tourné un T sur le côté. Si on applique le noyau [[0,0,1],[1,1,1],[0,0,1]] (T sur le côté) à l'image carre_orig.tif, on obtient la même image que carre_flou.tif. Donc on peut conclure que ça c'est le noyau de convolution.

Faites varier le paramètre lambda et commentez les résultats.

Lorsque nous augmentons la valeur du paramètre, le résultat s'améliore (Bruit sous-estimé), jusqu'à atteindre le bon résultat. À partir de là, lorsque nous augmentons le paramètre, le résultat s'aggrave (Bruit sur-estimé).

5. Applications

5.1. Comparaison filtrage linéaire et médian

```
im=skio.imread(r'C:\Users\jutogashi\Documents\Telecom_2A\IMA\IMA201\images_IMA201_filt\carre_orig.tif')
im=noise(im,5)
x=var_image(median_filter(im,typ=2,r=4),0,0,1,1)
for i in range(0,100):
    y=var_image(filtre_lineaire(im,get_cst_ker(i)),0,0,1,1)
    if(np.around(y,2)==np.around(x,2)):
        print(i)
```

i=1

5.2. Calcul théorique du paramètre de restauration

```
def wiener_mod(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

    imt=np.float32(im.copy())
    (ty,tx)=im.shape
    pi=np.pi
    aft=np.fft.fftshift(abs(np.fft.fft2(imt)))
    a=aft**2
    b=im.var()*im.size
    x=b/a

    #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+x)
    #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(r'C:\Users\jutogashi\Documents\Telec
om_2A\IMA\IMA201\images_IMA201_filt\carre_flou.tif'
)
viewimage(wiener(im,get_gau_ker(2),17))
viewimage(wiener_mod(im,get_gau_ker(2),17))
```