Compte Rendu TP2

BAHOUS Youness

October 1, 2022

1 Transformation géométrique :

- La méthode à plus proche voisin fait l'hypothèse que **l'image soit constante par morceaux** (ainsi le pixel porte la valeur du pixel connu auquel il est le plus proche) par contre pour la méthode bilinéaire , on suppose que l'image est **linéaire sur les deux dimensions** (ainsi donne au pixel inconnu la valeur calculée par interpolation linéaire sur les deux dimensions)

(a) Interpolation PPV

(b) Interpolation bilinéaire

Figure 1: Résultat de rotation

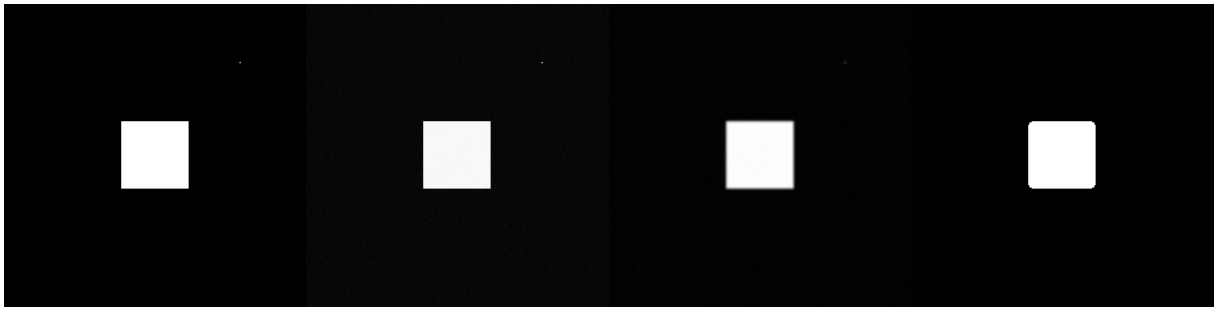
- Pour une image qui a subit 8 rotations de 45° , on remarque pour la méthode du PPV , on a perdu quelques pixels lors des interpolations grossières successives , on remarque que le chapeau et le bras de lena ont subi une faible transition. Pour la transformation bilinéaire je remarque que aprés les rotations successives on obtiens la quasi meme image que celle du départ
- Après avoir appliqué la rotation avec un zoom de facteur 1/2 , on remarque qu'on obtiens de l'aliasing sur les bords , c'est bien visible sur le bras de lena , ainsi il fallait appliquer un filtrage passe bas avant de sous échantillonner l'image.

Figure 2: Effet du dézoom sans échantillonnage

2 Filtrage linéaire et médian

- On remarque que la taille du noyau renvoyé par get_gau_ker augmente avec l'écart type donné en argument , et ceci s'explique par le simple fait qu'on veut rendre la gaussienne visible , car si on renvoie un petit tableau seules les valeurs proches du centre (éventuellement constantes) seront dans le noyau
- Pour quantifier le bruit résiduel dans une telle image il suffit de calculer la variance de d'une zone homogène de l'image filtrée , on remarque que à chaque fois qu'on augmente la variance du noyau du filtrage cette variance augmente , on remarque que la quantité du bruit dans une zone homogène est bien inférieur à celle obtenue par filtrage linéaire

```
im=skio.imread('../images/carre_orig.tif')
viewimage(im)
im_noisy = noise(im,2)
im_filtred_g =filtre_lineaire(im_noisy,get_gau_ker(1))
im_filtred_m=median_filter(im,r=3)
print(var_image(im_filtred_g,0,0,100,100))
print(var_image(im_filtred_m,0,0,100,100))
>> 14.152327472932784
>> 6.373750183799505
```

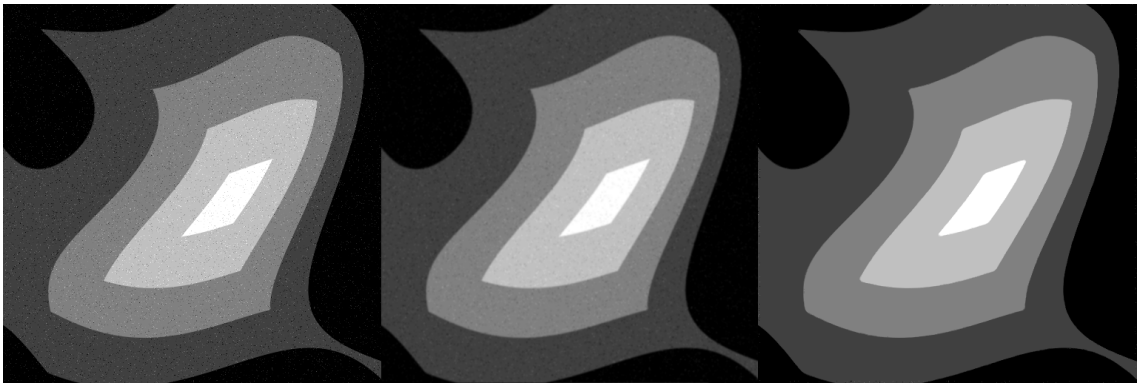


(a) Image originale (b) Image bruitée par un bruit de loi $\mathcal{N}(0,2)$ (c) Image filtrée à l'aide d'un filtre linéaire (d) Image filtrée à l'aide d'un filtre médian

Figure 3: Filtrage linéaire et médian - Comparaison

- Pour l'image pyra-impulse.tif :

On remarque que le filtrage linéaire détruit l'image , car en essayant d'éléminer le bruit il rends toutes les zones discontinues (contours par exemple) floues . par contre le filtrage médian conserve les contours mais déruit toutefois les entroits pointus sur le motif.



(a) Image originale (b) Image filtrée à l'aide d'un filtre linéaire (c) Image filtrée à l'aide d'un filtre médian

Figure 4: Filtrage linéaire et médian - Comparaison

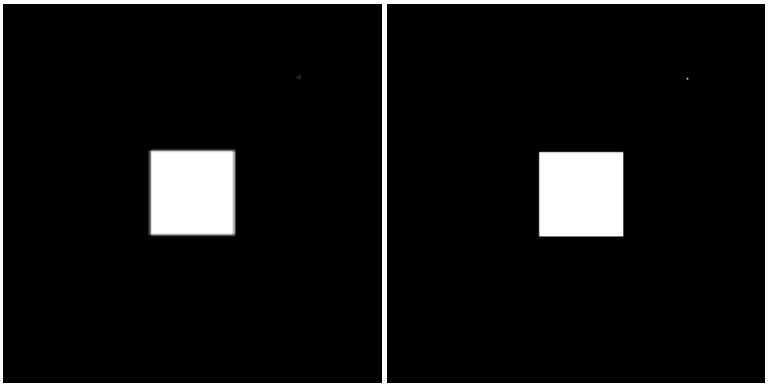
Pour le point séparé sur carre orig.tif , on remarque que le filtre médian le supprime , tandis que le filtrage linéaire le transforme en tache un peu plus sombre (effet de convolution d'un dirac avec une gaussienne)

3 Restauration

- En appliquent le filtrage linéaire inverse directement aprés le filtrage on obtient l'image du départ , mais si on ajoute un faible bruit entre les deux opérations ceci détruit complétement le contenu de l'image :

Figure 5: Image obtenue aprés filtrage inverge d'une image filtrée auquelle on ajouté un faible bruit

- pour déterminer le noyau de convolution qu'a subi l'image carre flou.tif on utiliser le point lumineux , en zoumant sur ce point on trouve qu'il corresponds au noyau $K = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$
- Par application du filtrage de Wiener : on ramarque que pour des λ faibles l'image n'est pas assez bizn restauré au fur et à mesure q'on augmente λ la qualité s'améliore , mais aprés une certaine valeure elle le filtre commence à déteriorer l'image



(a) carre_flou.tif (b) Image obtenue par filtrage Wiener

Figure 6: Effet du filtrage Wiener

4 Applications:

Comparaison filtrage linéaire et médian

```
1 im=skio.imread('../images/carre_orig.tif')
2 im = noise(im,5)
3 imfm = median_filter(im , r=4)
4 def imfc(im ,k):
5     imfc = filtre_lineaire(im,get_cst_ker(k))
6     return imfc
7 k= np.argmin([abs(var_image(imfc(im,k),0,0,100,100)-var_image(imfm,0,0,100,100)
    ) for k in range(100)])
8 print(k)
```

```
>> 6
```