TP filtrage non linéaire

Estimation du bruit

Nous avons implanté l'algorithme d'estimation du bruit (fonction estimer_bruit()). Pour l'image formes2bb90.pgm suivante par exemple, nous obtenons 88.140000 avec 15 et 0.5 en paramètres de l'algorithme. Cela est proche de la valeur de bruit indiqué.

Les Différent filtres :

Nous avons dans ce tp implémenté 4 différents types de filtres non linéaires, il s'agit de fonctions disponibles dans le fichier median.c.

I) Choix des paramètres :

I)1) filtre median

Exemple sur formes2bb90.pgm :

Figure 1: formes2bb90.pgm non filtre

On peut voir dans les exemples suivants l'effet de ce paramètre sur le filtrage : Nous utilisons comme exemple l'image formes2gbb90.pgm, le graphe des psnr ainsi que les resultats obtenus se trouvent à la page suivante.

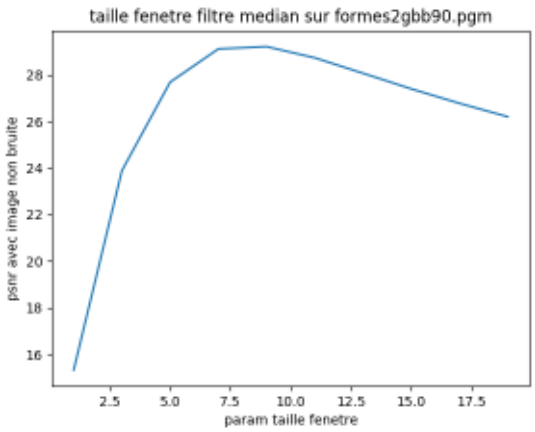


Figure 2:

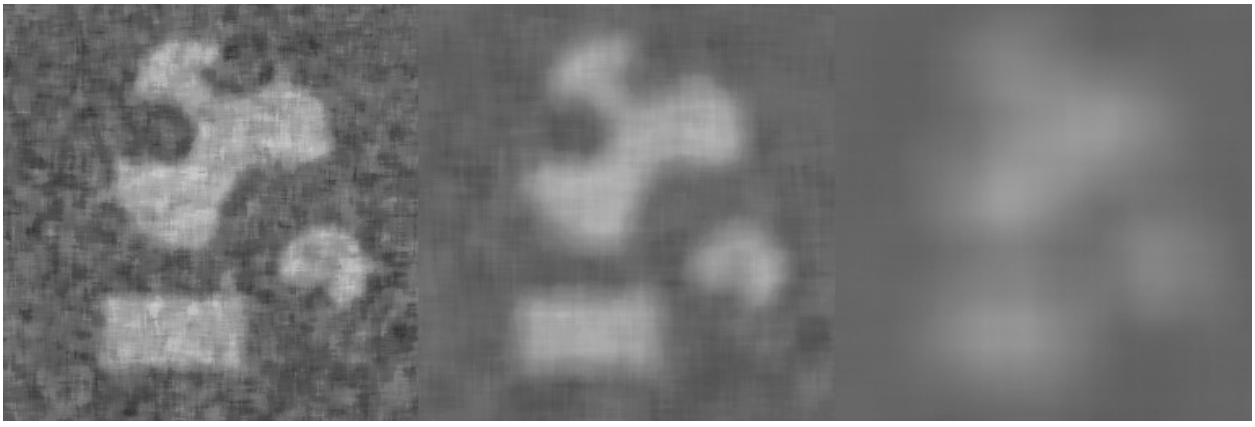


Figure 3: gauche-droite : t_fen = 4, 10, 30 sur formes2gbb90.pgm

Une fenetre trop grande floute l'image et dans le cas extreme finit par donner une image uniforme. On regarde la courbe de psnr pour trouver un bon compromis, et on choisit $t_fen = 10$ ici. On voit ici qu'une petite fenetre ne rend pas bien compte de l'unformité des zones de couleurs.

I)2) -filtre bilateral

Exemple sur formes2bb90.pgm (image non filtre deja montree plus haut) Pour ce filtre, il faut choisir sigma1 ainsi que sigma2. En augmentant sigma1 on approche une gaussienne, en augmentant sigma2 on lisse mieux. courbe de psnr suivant sigma1 a sigma2 fixé et vice et versa sur un exemple?

On a appliqué un lissage median comme suggére dans le tp, on notera que on utilise pour cela le filtre median précédent avec une fenêtre de taille 3.

Comme prévu sigma1 augmente entraine un meilleur psnr et conservation de contour cependant cela augmente la complexité du filtre rapidement. On prendra $sigma1 = 5$ ainsi que $sigma2 = 80$ cela fournit de bonnes valeurs pour le psnr et assez bien pour les contours.

On voit bien ce resultat sur les images ci-dessous

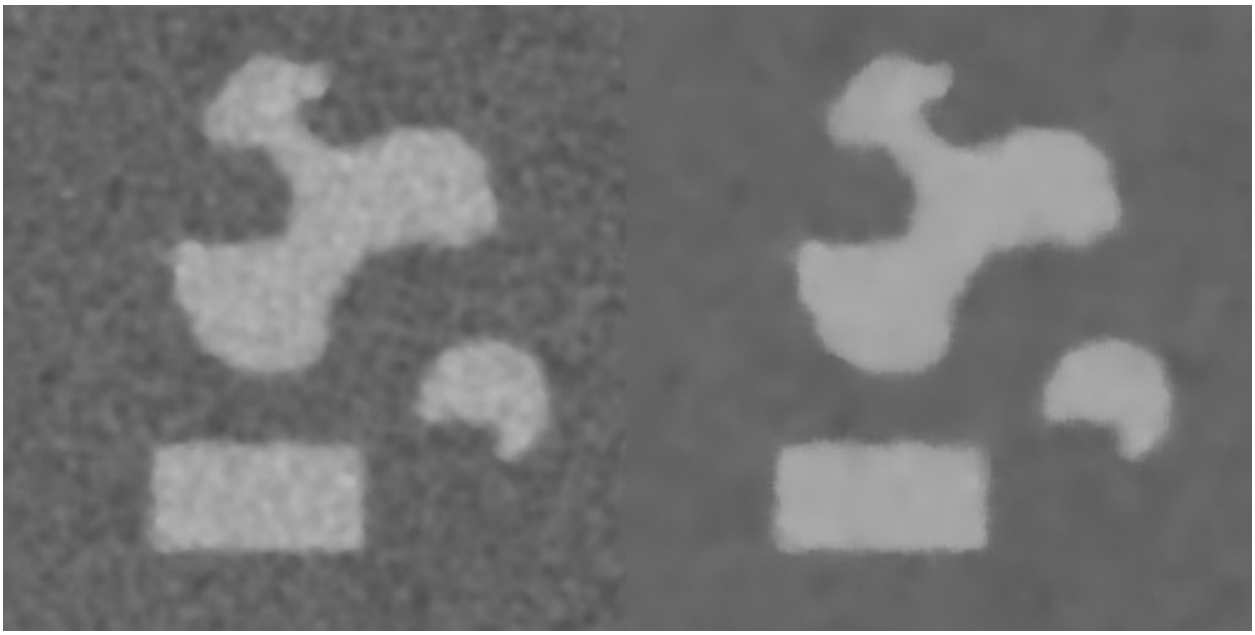


Figure 4: Influence de sigma1, de 2 à gauche et 5 à droite, sigma2=20 fixé

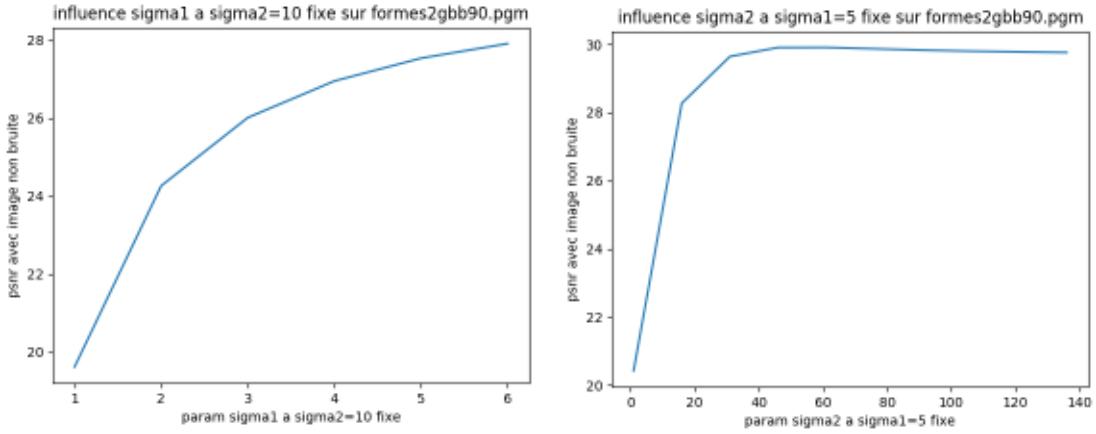


Figure 5:

I)3) -filtre adaptatif recursif

Exemple sur formes2gbb40.pgm :

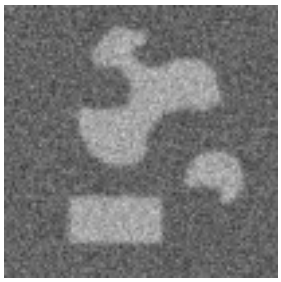
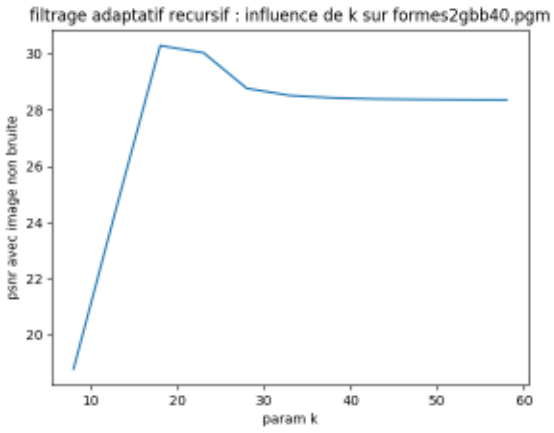


Figure 6: non filtre

Pour ce filtre il faut choisir k. Il est lié au bruit de l'image car correspond environ au seuil de conservation du gradient.



Nous avons choisi de le prendre environ égal à la mesure du bruit. Par exemple sur l'image formes2bb40.pgm nous prendrons k du meme ordre de grandeur ($bruit/2$ semble donner de bons resultats). On peut également,

si la valeur du bruit n'est pas connue, la calculer en utilisant la fonction $estimer_bruit()$ que nous avons implanté et utiliser cette valeur pour k, nous trouvons sur cette image une valeur de bruit de 45.20.

Un k trop grand floute trop l'image. On retient ici $k = 20$ qui minimise le psnr, on remarque que cela correspond a $bruit/2$.

Pour cette valuer on obtient aussi une bonne conservation des contours (cela correspondant à l'image du milieu sur la page suivante), même si certaines zones sont légèrement déjà floutées, on pourrait affiner pour chercher un k encore meilleur (un peu plus petit).

La premiere image (k=13) ne fournit quand à elle pas a un resultat très satisfaisant, et l'uniformité des couleurs n'est pas conservée.

Les figures page suivante illustrent cette analyse.

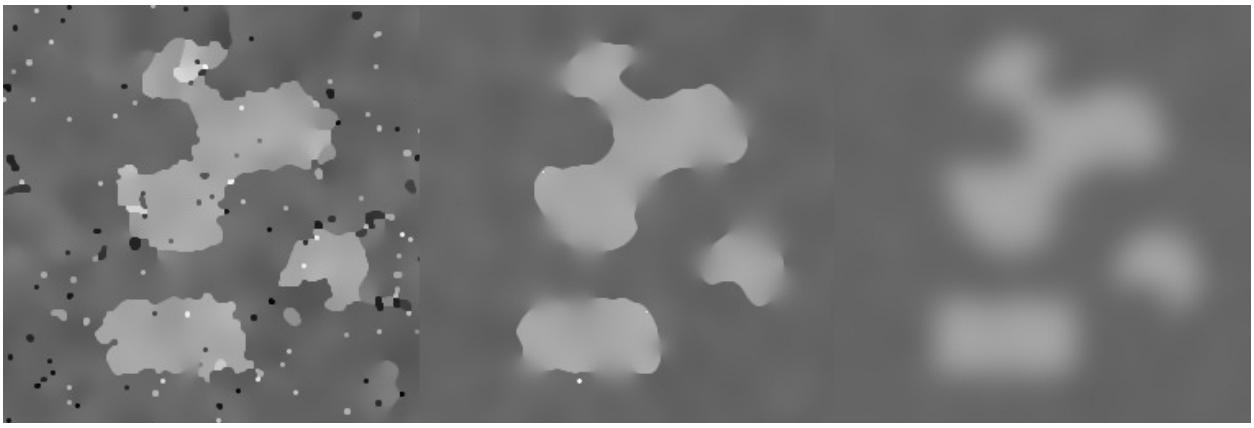


Figure 7: De gauche à droite : k = 13, 20, 48 sur formes2gbb40.pgm

I)4) filtre NI_mean

Exemple sur formes2bb90.pgm

Figure 8: non filtre a gauche, filtre a droite

Le tableau fournit en fin de tp nous donne les tailles des patch des regions ainsi que la valeur du paramètre h en fonction de sigma. Il suffit donc de regarder le sigma qui convient pour choisir les bons paramètres.

Par exemple sur l'image bruité formes2bb90.pgm (déjà affichée plus haut), on teste les possibilités et on en déduit l'intervalle ou se trouve sigma. On obtient :

Comparaison des psnr pour les 4 intervalles sur un exemple. On utilise le tableau fournit en tp pour déduire de sigma les autres parametres. On obtient : (sigma, psnr) les valeurs :

[(5, 10.475291), (20, 10.475562)] Nous avons un bug avec notre filtre NI_mean : celui-ci ne modifit que peu les images car tout les termes w_pq valent 0 (trop petits) sauf lorsque les deux termes dont egaux et alors w_pq = 1. On le voit dans l'image suivante (sigma = 20) :

Figure 9:

II) Comparaison des filtres psnr :

Graphes de psnr pour différentes filtres sur différents types de bruits. On choisit des paramètres bons pour les filtres en regardant psnr et en prenant en compte les remarques faites lors de la question précédente.

- bb (formes2bb90.pgm) (avec paramètres donnant de bons psnr)

Tableau des psnr :

Median : 29.207321

adaptif recursif : 30.096830

bilateral : 26.830254

NI-mean : 6.772923

D'apres le psnr, le filtre adaptatif recursif semble le plus efficace selon cette métrique. Mais on verra que néanmoins cela ne veut pas dire que les contours sont bien conservés (le psnr n'étant qu'une mesure de débruitage).

- gs (formes2gs017.pgm) (avec paramètres donnant de bons psnr)

Tableau des psnr :

Median : 30.081084

adaptif recursif : 33.628137

bilateral : 28.200811

NI-mean : 10.773768

On voit bien sur l'image suivant qu"un psnr elevé n'assure pas la conservation des contours, et c'est le filtrage bilateral et Median qui les conservent le mieux pour les valeurs des paramètres choisis pour maximiser le psnr. (Ce sont ces paramètres que nous avons utilisé ici).

On retrouve encore un classement similaire selon les psnr.

III) Temps d'execution :

- sur une image de type formes2gs017.pgm :

Figure 10: De gauche à droite : filtrage adaptatif, bilateral, Median pour gs2017

filtrage median : 0.070356

filtrage bilat : 5.261069

filtrage adaptatif recursif : 2.479394

filtrage niMean : 5.056486

On voit que le filtre le plus rapide est le filtre médian ainsi que le filtrage niMean. Le filtre le plus lent étant le filtre bilatéral. Le filtre bilatéral est lent car on a choisit une assez grande valeur de sigma1 (6) pour avoir un bon psnr. Ce qui augmente la complexité temporelle.

Le filtre adaptatif recursif est aussi plutot lent car on utilise la methode ou les w_t change pour tout t, et non w_t = w_0 ce qui irait plus vite mais serait moins précis.

Filtrage sur d'autres images

Exemple de tangram.pmg: (resultats page suivante)

On remarquera encore que le filtre NI_mean ne fait pas ce qu'il devrait en raison de notre problème déjà mentionné plus haut. Tangram :

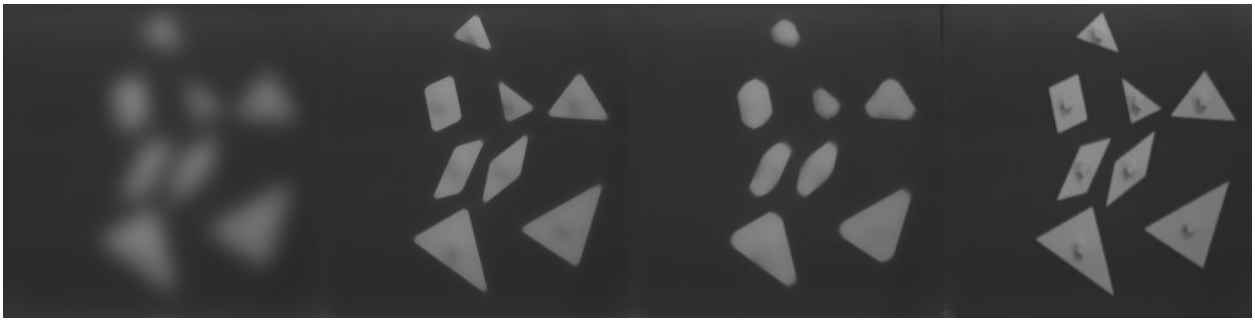


Figure 11: de gauche a droite : adaptatif recursif, bilateral, Median, Ni_mean

Le Bilatéral, avec les paramètres choisis ici est celui qui a le mieux conservé les contours avec le Median, l'adaptatif a débruité mais a flouté les contours car les paramètres n'étaient pas assez bon.