Exercice Chapitre 3 – Filtrage linéaire

Dans cet exercice, vous allez réaliser un filtrage linéaire sous Matlab de différentes manières. Avant de commencer, chargez et décompressez le fichier archive « ***filtreLineaire.zip***» qui contient les scripts nécessaires tout au long de cet exercice.

Filtrage linéaire dans le domaine fréquentiel et dans le domaine spatial

1 – Ouvrez le script ***filtrage_passebas.m***. À l'aide de la touche F9 lancez la première partie du script (calcul de la fonction de transfert du filtre). Changez la taille du support du filtre et relancer le script avec F9. Expliquez les résultats obtenus.

2 – On réalise, dans la deuxième partie, le filtrage spatial d'une image en utilisant la fonction ***conv2*** de Matlab (convolution 2D) selon deux manières différentes. Comparez les deux images résultats et concluez (vous pouvez changer le support du filtre, dans ce cas il faut évidemment relancer la première partie du script).

3 – On utilise dans la troisième partie du script, la fonction ***imfilter*** de Matlab de deux façons différentes. Comparez les pixels sur les bords à droite et concluez. Après avoir observé les effets du filtrage en spatial, observez le résultat en fréquentiel à l'aide de la dernière partie du script.

4 – Exécutez le script ***filtrage_spatialvsfreq.m***. Celui-ci permet de comparer le temps d'exécution entre un filtrage spatial et un filtrage en fréquentiel. Jouez sur la taille des supports et concluez.

5 – Analysez le script ***filtrage_passehaut.m***. Exécutez le et analysez les résultats obtenus.

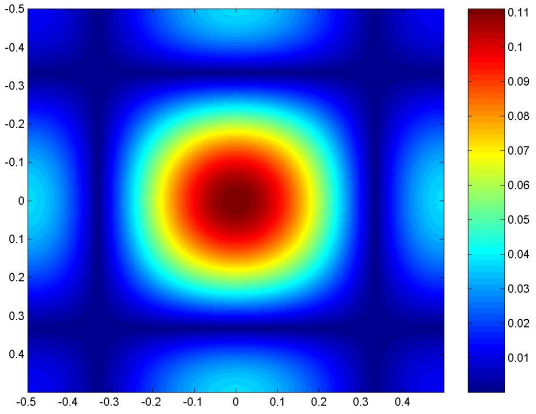
Correction de l'exercice : Filtrage linéaire

Filtrage linéaire dans le domaine fréquentiel et dans le domaine spatial

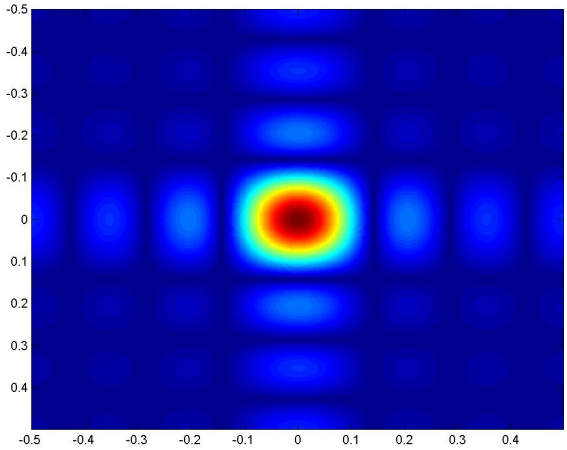
1 – La première partie du script ***filtrage_passebas.m*** définit le noyau du filtre passe bas qui sera utilisé. Dans le cas présent, il s'agit du noyau d'un filtre moyenneur qui donne à chaque pixel la valeur moyenne de son voisinage :

$$\frac{1}{9}.\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Le module de la fonction de transfert de ce filtre est ensuite affiché :

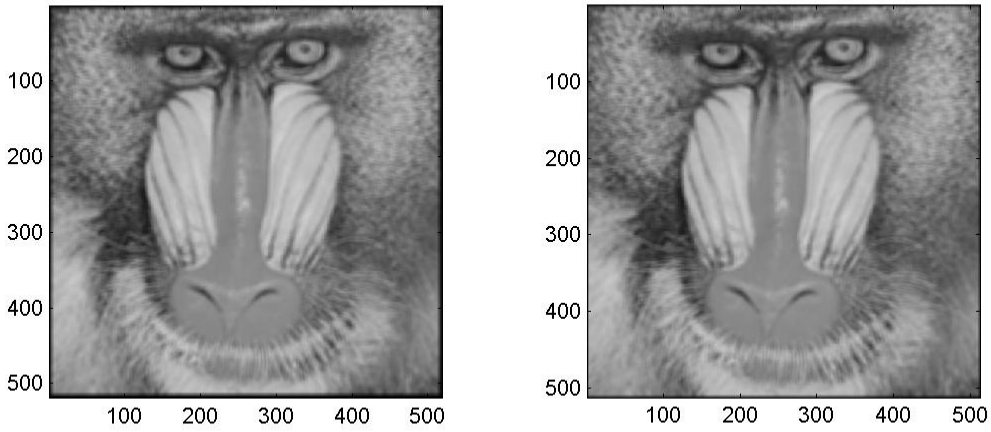


Sur l'image, on observe que le filtre moyenneur laisse passer les basses fréquences. Cependant, le gain remonte sur l'intervalle des moyennes et hautes fréquences spatiales normalisées. Le filtre passe-bas n'est donc pas très sélectif.
En augmentant la taille du support du filtre (support=7), on obtient le module de fonction de transfert suivant :



En augmentant le support, on observe que la bande passante du filtre est réduite : le filtre est plus sélectif. Le gain a moins tendance à remonter pour les moyennes et les hautes fréquences normalisées. En fait, chaque pixel est la valeur moyenne de son voisinage, donc plus le support du filtre est grand, plus le voisinage est grand et donc les pixels ont, après filtrage, des valeurs proches les unes des autres (effet de lissage). Les changements de niveaux selon (Ox) ou (Oy) sont donc lents (faibles variations de luminance) et les fréquences associées sont donc basses.

2 – En lançant la deuxième partie du script ***filtrage_passebas.m***, on obtient la figure suivante :



Ces images sont obtenues avec un filtre de taille 7x7. Le filtrage est ici réalisé en calculant la convolution « filtre⊗image » avec la fonction ***conv2*** de Matlab. Un effet de lissage est nettement visible sur les deux images. Cependant on remarque dans le workspace, que la taille de l'image de droite (imf2) est inférieure à la taille de l'image de gauche (imf). En effet, ces deux images sont calculées avec la même fonction ***conv2*** mais avec des paramètres d'entrée différents :

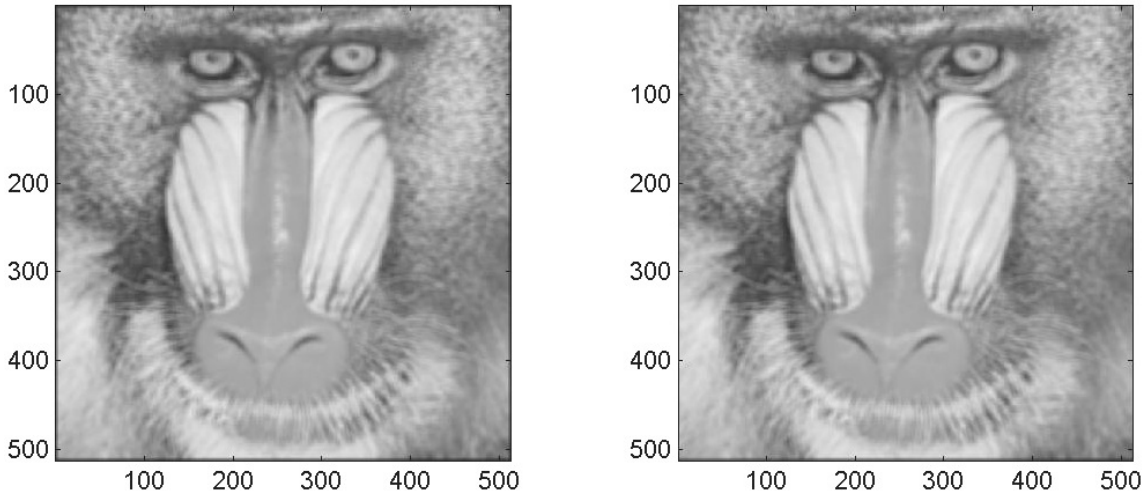
```
imf = conv2(im1,filtre); % (équivalent à imf=conv2(im1,filtre,'full');)
imf2 = conv2(im1,filtre,'same');
```

Si on considère une image d'entrée de taille M_1 x N_1, et un filtre de taille M_2 x N_2 :

- L'image imf est calculée en réalisant le produit de convolution de deux signaux 2-D. Elle a donc une taille (M_1+M_2-1) x (N_1+N_2-1).
- L'image imf2 est calculée en réalisant le même produit de convolution mais en ne conservant que la partie centrale du résultat de façon à obtenir en sortie une image de la même taille que l'image d'entrée.

Notons qu'il existe également un troisième paramètre ('valid') pour la fonction ***conv2***. Dans ce cas, la sortie est composée des éléments de la convolution qui ont été calculés en supprimant les effets de bord. L'image de sortie a donc une taille (M_1-M_2+1) x (N_1-N_2+1).

3 – En lançant la troisième partie du script ***filtrage_passebas.m***, on obtient la figure suivante (support du filtre 7x7) :



Ces deux images sont obtenues grâce à la fonction ***imfilter*** de Matlab :

```
imf3 = imfilter(im1,filtre);
imf4 = imfilter(im1,filtre,'replicate');
```

On remarque que le bord droit est plus sombre sur l'image de gauche. En fait, les pixels de bord dans une image n'ont pas de voisinage en dehors de l'image. Pour calculer la convolution, il faut donc choisir une méthode pour traiter ces pixels. Matlab permet d'utiliser plusieurs méthodes de traitement des bords : mise à 0 (méthode par défaut), duplication (`'replicate'`), symétrie (`'symmetric'`), ...

Exemples de traitements de bord :

Considérons l'image monochrome I de taille M x N suivante :

7	1	72	1	2	1
25	5	100	5	10	209
87	125	40	87	248	28
154	42	25	111	2	1
0	45	114	5	194	68
58	18	47	220	4	8

On désire filtrer cette image avec le filtre moyenneur « h » de support 3x3. Les pixels marqués d'une étoile rouge n'ont donc pas de voisinage en dehors de l'image.

- Mise à 0 du voisinage en dehors de l'image :

On peut par exemple considérer que le voisinage en dehors de l'image est constitué de pixels à 0. On peut ainsi effectuer la convolution sur les pixels de bord marqués d'une étoile rouge.

0	0	0	0	0	0	0	0
0	7	1	72	1	2	1	0
0	25	5	100	5	10	209	0
0	87	125	40	87	248	28	0
0	154	42	25	111	2	1	0
0	0	45	114	5	194	68	0
0	58	18	47	220	4	8	0
0	0	0	0	0	0	0	0

Il s'agit de la méthode par défaut utilisée par Matlab.

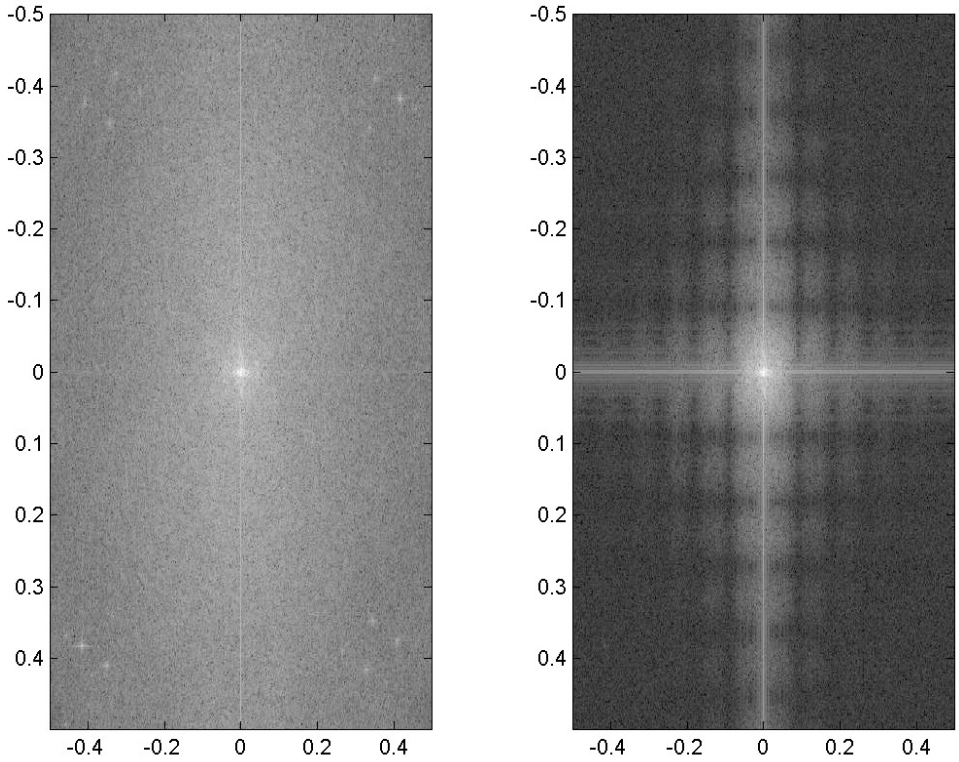
- Duplication :

Pour calculer la convolution, on peut également dupliquer les pixels de bords afin de créer un voisinage. Chaque pixel du voisinage, situé en dehors de l'image, prend donc la valeur du pixel le plus proche appartenant à l'image.

7	7	1	72	1	2	1	1
7	7	1	72	1	2	1	1
25	25	5	100	5	10	209	209
87	87	125	40	87	248	28	28
154	154	42	25	111	2	1	1
0	0	45	114	5	194	68	68
58	58	18	47	220	4	8	8
58	58	18	47	220	4	8	8

On remarque que dans le cas particulier d'un filtre de taille 3x3, cette méthode de duplication a le même effet qu'une autre méthode : la symétrie.

La dernière partie du script ***filtrage_passebas.m*** permet d'afficher, pour l'image *MANDRILL_LUMI.BMP*, le module du spectre avant et après filtrage :



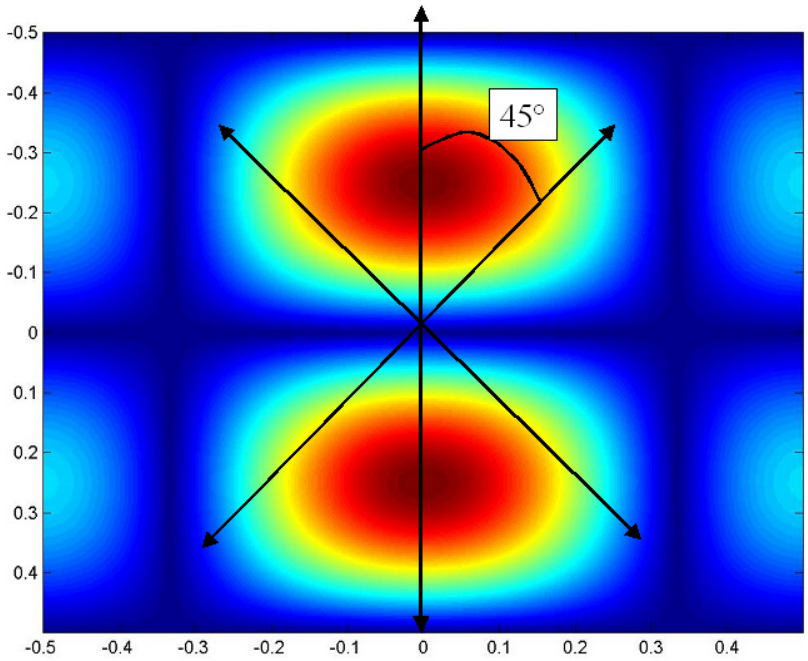
L'image à gauche représente le module du spectre de *MANDRILL_LUMI.BMP* avant filtrage. On aperçoit une zone claire au centre du spectre, qui correspond à la composante continue de l'image. Le nuage de points autour de cette composante continue correspond à quelques composantes moyennes et hautes fréquences de l'image (contours, détails, zones de transition, …).
L'image de droite représente le spectre de *MANDRILL_LUMI.BMP* après filtrage passe-bas. La composante continue qui est basse fréquence est bien conservée. Le nuage de points représentant les moyennes et hautes fréquences a été globalement supprimé. Cependant, selon les axes des fréquences spatiales horizontales et verticales pures, des gains importants apparaissent pour les moyennes et hautes fréquences. Ce résultat était prévisible après observation du module de la fonction de transfert du filtre moyenneur effectuée en 1.

4 – Le script ***filtrage_spatialvsfreq.m*** permet de comparer, grâce aux commandes ***tic*** et ***toc*** de Matlab, le temps d'exécution entre un filtrage spatial (calcul de convolution avec ***imfilter***) et un filtrage en fréquentiel (utilisation de FFT avec ***fft2***, ***fftshift*** et **ifft2**).
On observe alors que le temps d'exécution est moins important pour le filtrage en fréquentiel que pour le filtrage spatial lorsque le support du filtre est assez grand (plus grand que 17 dans le cas présent).
En effet, plus le support du filtre est grand, plus le calcul de la valeur d'un pixel par convolution nécessite d'opérations du type : « additions et multiplications ». A l'inverse, quelque soit la taille du support du filtre, le calcul de la valeur d'un pixel avec FFT ne nécessite toujours qu'une multiplication suivie d'une transformation (IFFT : inverse FFT).

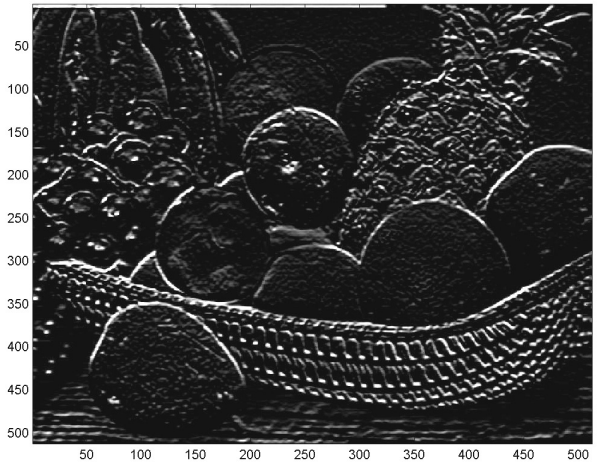
5 – La première partie du script ***filtrage_passehaut.m*** permet d'afficher le module de la fonction de transfert du filtre passe-haut que l'on souhaite utiliser. Ici, on considère le filtre de Prewitt vertical dont le noyau est :

$$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Une analyse rapide de la structure de ce noyau de convolution montre que la différence entre les valeurs des pixels sera augmentée selon la direction verticale et diminuée selon la direction horizontale. Le filtre sera donc passe-haut pour les fréquences verticales. On observe très bien cette caractéristique sur l'image du module de la fonction de transfert de ce filtre :



On remarque toutefois que le filtre n'est pas très sélectif, ni en fréquences spatiales, ni en orientation (tolérance sur les contours d'inclinaisons à ± 45°).
En filtrant l'image *FRUIT_LUMI.BMP* avec ce filtre, on obtient l'image :

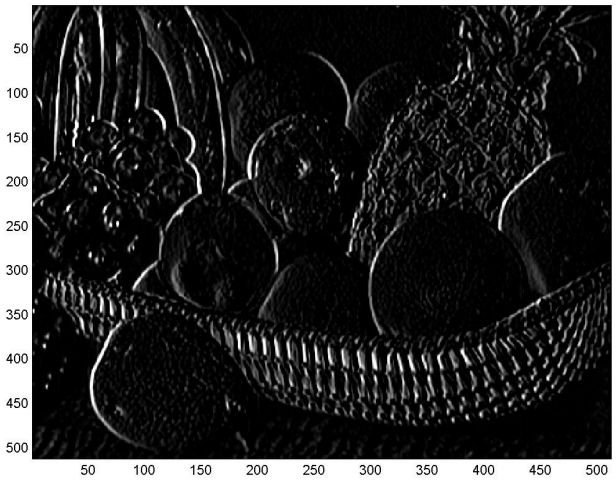


On a obtenu la détection des contours verticaux de l'image.

Un second filtre est proposé, il s’agit du filtre Prewitt horizontal dont le noyau est :

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Ce filtre permet de détecter les contours horizontaux :



De tels filtres sont généralement utilisés pour renforcer les contours dans une image. Typiquement, on associe l’image de référence à l’image filtrée.