Chapitre 3

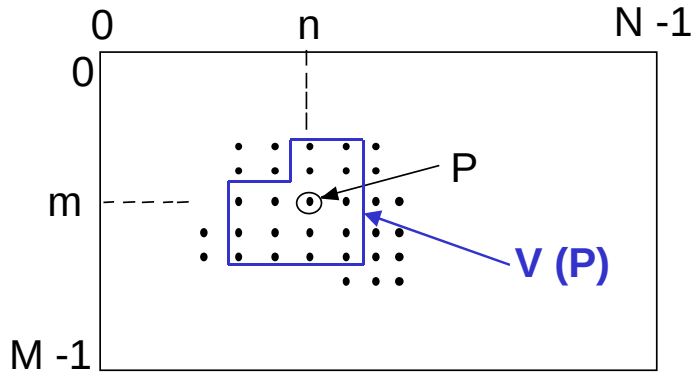
Fondamentaux du traitement d'image

Filtrage Linéaire

Voisinage d'un point

Notion de voisinage d'un pixel

- Point P d'affixe p = (m, n)
- Son voisinage : V(P) = {P' connectés à P}



Domaine discret de l'image

Exemples: 4-N  8-N 

4-connexité 8-connexité

Le traitement d'image s'appuie fondamentalement sur des traitements à l'aide de voisinages. Cela signifie que les traitements effectués en un endroit donné correspondant à un pixel dépendent non seulement de ce pixel mais aussi de pixels appartenant à son voisinage.
Considérons un pixel P dont la position dans l'image est donnée par les coordonnées (m, n). Son affixe est donc p = (m, n). Un voisinage de P, noté V(P), se définit comme un ensemble de pixels P' connectés à P. Le pixel P, cerclé dans la figure, appartient à son propre voisinage V(P).
Il faut donc, à ce niveau, définir la notion de « composantes connexes » au sein d'une image discrète, afin de savoir comment des pixels sont connectés entre eux.
Plutôt que de s'étendre sur la notion de connexité, nous présentons ici des figures illustrant les deux cas les plus couramment choisis :

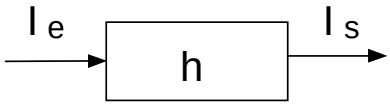
- un voisinage de « 4-connexité » : le pixel (centré et entouré dans la figure) n'a que quatre pixels voisins, chacun d'eux étant à distance unité de P avec la distance d_4 ;
- un voisinage de « 8-connexité » : le pixel (au centre et entouré) a huit pixels voisins, chacun d'eux étant également à distance unité de P, mais avec la distance d_8.

On définit les deux distances suivantes en voisinage numérique à structure d'échantillonnage carrée :

- $d_4(P, P') = | m - m' | + | n - n' |$;
- $d_8(P, P') = \mathrm{Sup}(| m - m' |, | n - n' |)$.

Opérations sur un voisinage

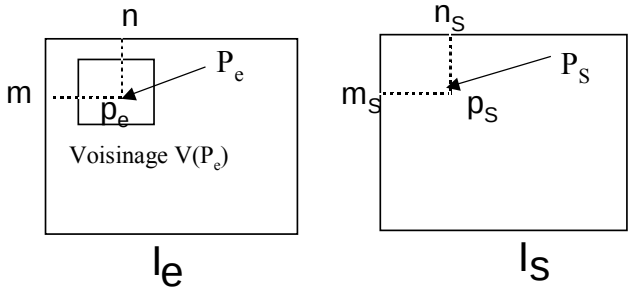
Filtrage linéaire (convolution)



Filtrage « transversal »: h est la réponse impulsionnelle 2-D appelée aussi « fonction d' étalement du point »

➢ Filtre à réponse impulsionnelle finie – RIF

$$I_S(m,n) = \sum_{(i,j)\in V(P)} \sum h(i,j)\, I_e(m-i\ ,n-j)$$



➢ Filtre récursif – IIR

$$Is(m,n)=\frac{1}{b_{0,0}}\times\left(\sum_{(i',j')\in V(P)}\sum a_{i',j'}.Ie(m-i',n-j')-\sum_{(i,j)\in V'(P')}\sum b_{i,j}.Is(m-i-1,n-j-1)\right)$$

Le principe est de construire à partir d'une première image I_e, une seconde image I_S généralement de même taille. Chaque pixel P_S d'affixe $p_S = (m_S, n_S)$ de l'image de sortie I_S est calculé à partir des pixels du voisinage $V(P_e)$ du point P_e d'affixe $p_e = (m, n)$ de l'image I_e. Usuellement $p_S = p_e$, i.e. $m = m_S$, et $n = n_S$.

Dans le cas le plus simple, la valeur du pixel P_S est obtenue par une combinaison linéaire des valeurs des pixels de $V(P_e)$, on parle d'opération de convolution (filtrage linéaire).

On donne ici l'exemple d'un filtrage transversal où la valeur Is(m, n) du pixel P_S est une combinaison linéaire des valeurs $I_e(m, n)$ des pixels du voisinage de P_e, pondérées par les coefficients h(i, j) de la réponse impulsionnelle du filtre :

$$I_S(m,n)=\sum_{(i,j)\in V(P_e)}\sum h(i,j).I_e(m-i,n-j)$$

La taille du voisinage est donc définie par la taille du support de la Réponse Impulsionnelle Finie (RIF) « h » du filtre.

Notons qu'il existe également des filtres récursifs. Le calcul des valeurs des pixels de I_S est alors plus complexe car la combinaison linéaire tient compte :

- du voisinage de I_e (m, n), le jeu de coefficients du filtre associé est l'ensemble $\{a_{i',j'}\}$;
- des valeurs antérieures de Is(m, n) précédemment calculées (récursivité), le jeu de coefficients du filtre associé est alors l'ensemble $\{b_{i,j}\}$.

$$I_S(m,n)=\frac{1}{b_{0,0}}.\sum_{(i',j')\in V(P_e)}\sum a_{i',j'}.I_e(m-i',n-j')-\sum_{(i,j)\in V(P_S)}\sum b_{i,j}.I_S(m-i-1,n-j-1)$$

Le résultat dépend donc de la méthode choisie pour parcourir l'image Ie (notion de causalité) car cela influe directement sur les valeurs antérieures I_S (m, n).

Exemple – Convolution – Corrélation

Exemple : filtre de support de taille (3×5) (3 en direction verticale et 5 en horizontale)

Noyau de convolution: $h = \begin{bmatrix} h_{-1,-2} & h_{-1,-1} & h_{-1,0} & h_{-1,1} & h_{-1,2} \\ h_{0,-2} & h_{0,-1} & h_{0,0} & h_{0,1} & h_{0,2} \\ h_{1,-2} & h_{1,-1} & h_{1,0} & h_{1,1} & h_{1,2} \end{bmatrix}$

Convolution $$I_s(m,n) = \sum_{j=-2}^{+2} \sum_{i=-1}^{+1} h(i,j).I_e(m-i,n-j)$$

Corrélation Soit $h^*(i,j) = h(-i,-j)$

(=> symétrique de h par rapport à (0,0))

$$I_s(m,n) = \sum_{j=-2}^{+2} \sum_{i=-1}^{+1} h^*(i,j).I_e(m+i,n+j)$$

$$h^* = \begin{bmatrix} h_{1,2} & h_{1,1} & h_{1,0} & h_{1,-1} & h_{1,-2} \\ h_{0,2} & h_{0,1} & h_{0,0} & h_{0,-1} & h_{0,-2} \\ h_{-1,2} & h_{-1,1} & h_{-1,0} & h_{-1,-1} & h_{-1,-2} \end{bmatrix}$$

Dans le cas d'une convolution, le filtre est entièrement caractérisé par l'ensemble des coefficients { h(i,j) }, noté aussi { $h_{i,j}$ } pour rappeler que i et j sont des variables discrètes. On appelle cet ensemble le « noyau » du filtre.
De fait, ce noyau définit à la fois :

- le voisinage $V(P_e)$ à utiliser (dans l'exemple il s'agit d'un voisinage de taille (3×5) où la position (0,0) est à centrer sur P_e) ;
- les poids respectifs (les valeurs des h(i, j)) de chaque pixel de ce voisinage, afin de calculer la nouvelle valeur P_S.

La taille du support rectangulaire (I, J) et les poids étant connus, il est alors possible de calculer tous les pixels de l'image Is grâce à la relation :

$$I_S(m,n) = \sum_{i \in I} \sum_{j \in J} h(i,j).I_e(m-i,n-j)$$

Remarque : il faut prendre soin de définir des « conditions aux bords », c.a.d proposer une solution particulière afin de traiter les pixels aux bords de l'image. En effet, ces derniers n'ont pas de voisins à l'extérieur de I_e (une solution simple est de ne retenir que les pixels existants). Diverses méthodes de traitement des bords sont présentées dans l'exercice : « Filtrage Linéaire» de ce même chapitre.

A titre d'exemple, on considère un filtre « h », dont le noyau de convolution est :

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Ce filtre est en fait la somme d'un filtre Laplacien (détection de contours) et d'un filtre Identité, l'ensemble forme donc un filtre rehausseur de contours (*enhancement*).

On considère une image I_e de taille 3×3 :

$$\begin{bmatrix} 4 & 125 & 255 \\ 7 & 0 & 45 \\ 9 & 56 & 13 \end{bmatrix}$$

L'image I_e est filtrée par h. Voici le détail du calcul pour déterminer, après filtrage, la valeur du pixel central $I_S(1, 1)$:

$$I_S(1,1) = \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} h(i,j) \,.\, I_e(1-i,1-j)$$

$$= \sum_{i=-1}^{+1} h(i,-1) \,.\, I_e(1-i,1-(-1)) + h(i,0) \,.\, I_e(1-i,1-0) + h(i,1) \,.\, I_e(1-i,1-1)$$

$$= \Big[h(-1,-1).I_e(1+1,2) + h(-1,0).I_e(1+1,1) + h(-1,1).I_e(1+1,0)$$

$$+ h(0,-1).I_e(1,2) + h(0,0).I_e(1,1) + h(0,1).I_e(1,0)$$

$$+ h(1,-1).I_e(1-1,2) + h(1,0).I_e(1-1,1) + h(1,1).I_e(1-1,0) \Big]$$

$$= (0\times13) + (-1\times56) + (0\times9) + (-1\times45) + (5\times0) + (-1\times7) + (0\times255) + (-1\times125) + (0\times4)$$

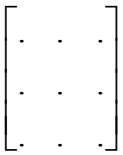
$$I_S(1,1) = -233$$

<u>Mise en garde</u> : la numérotation des lignes et des colonnes n'est pas la même pour l'image d'entrée et le filtre. Pour l'image, les indices vont de 0 à M-1 (lignes) et de 0 à N-1 (colonnes). Pour le filtre, l'élément $h_{0,0}$ est placé au centre du noyau de convolution, les indices vont donc de –I à +I (lignes) et de –J à J (colonnes).

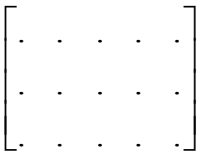
Notons que, si l'on considère la fonction bidimensionnelle « h^* » symétrique de « h » par rapport à (0, 0), on peut également écrire la valeur du pixel de l'image de sortie I_S comme étant la corrélation entre I_e et h^* :

$$I_S(m,n) = \sum_{i\in I} \sum_{j\in J} h^*(i,j).I_e(m+i,n+j)$$

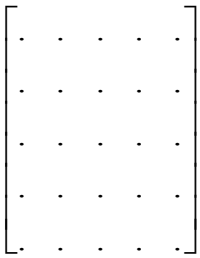
Exemples de masques de convolution typiques



Masque 3×3



Masque 3×5



Masque 5×5

Masques de convolution plus larges :

(~ circulaire: 45 pixels ; rectangulaire: 11x11)

• Gain en continu « DC » du filtre : $\sum_i \sum_j h(i, j) = H(\nu_X = 0, \nu_Y = 0)$

• Filtre symétrique : $h(-i,-j) = h(i,j)$ alors : convolution ≡ corrélation

=> $H(\nu_X, \nu_Y)$ est une fonction de transfert réelle (pas de terme de phase)

On peut imaginer toutes sortes de masques de convolution. La taille de ce dernier définit la taille du voisinage. Différents supports pour le noyau sont présentés ici : 3×3, 3×5, et 5×5.
Comme dans le cas des signaux 1-D, on définit la réponse fréquentielle $H(\nu_X, \nu_Y)$ du filtre qui évolue en fonction des fréquences spatiales horizontales (ν_X) et verticales (ν_Y).
$H(\nu_X, \nu_Y)$ est la transformée de Fourier 2-D de la réponse impulsionnelle « h » :

$$H(\nu_X, \nu_Y) = \sum_n \sum_m h(m, n) \exp\left[-2j\pi(n\nu_X + m\nu_Y)\right]$$

On peut montrer que le gain en continu (gain de la réponse fréquentielle H aux fréquences spatiales nulles : $\nu_X = 0$, $\nu_Y = 0$) du filtre se calcule comme la somme de tous les coefficients h(i, j) du noyau : $\text{Gain\_DC} = \sum_i \sum_j h(i,j)$.

L'exercice « FFT » présente plus en détail les particularités et les méthodes de représentation pour les spectres d'images et les réponses fréquentielles des filtres. La notion de fréquences spatiales y est approfondie.
Dans le cas particulier où le filtre est symétrique, $H(\nu_X, \nu_Y)$ est une réponse fréquentielle réelle.

Remarque : dans la suite du chapitre, par abus de langage, on parlera de fonction de transfert d'un filtre plutôt que de réponse fréquentielle.

Exercice Chapitre 3 – Convolution

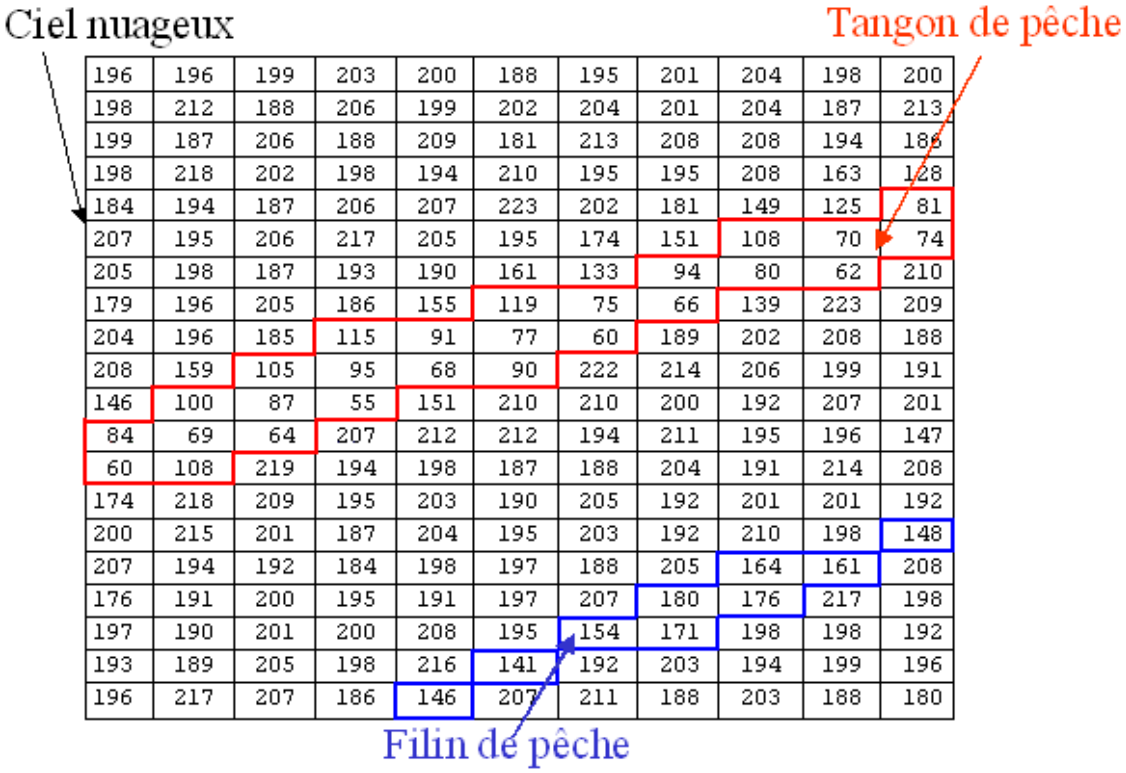
Dans cet exercice, aucune programmation n'est demandée. Le but est d'effectuer, à la main, un filtrage en calculant le produit de convolution de l'image d'entrée avec le noyau d'un filtre donné.

Produit de convolution

Nous allons appliquer l'opérateur de convolution sur une petite portion de l'image « *BOATS* ». Sur la figure ci-dessous, cette portion est délimitée par le rectangle rouge contenant une partie du tangon du bateau de pêche.

Image « BOATS »

La matrice I_0 suivante contient les valeurs de luminance des pixels pour la zone (20×11) délimitée par le rectangle rouge. On distingue facilement les pixels appartenant au tangon et au filin de pêche dans deux bandes diagonales de la matrice.



196	196	199	203	200	188	195	201	204	198	200
198	212	188	206	199	202	204	201	204	187	213
199	187	206	188	209	181	213	208	208	194	186
198	218	202	198	194	210	195	195	208	163	128
184	194	187	206	207	223	202	181	149	125	81
207	195	206	217	205	195	174	151	108	70	74
205	198	187	193	190	161	133	94	80	62	210
179	196	205	186	155	119	75	66	139	223	209
204	196	185	115	91	77	60	189	202	208	188
208	159	105	95	68	90	222	214	206	199	191
146	100	87	55	151	210	210	200	192	207	201
84	69	64	207	212	212	194	211	195	196	147
60	108	219	194	198	187	188	204	191	214	208
174	218	209	195	203	190	205	192	201	201	192
200	215	201	187	204	195	203	192	210	198	148
207	194	192	184	198	197	188	205	164	161	208
176	191	200	195	191	197	207	180	176	217	198
197	190	201	200	208	195	154	171	198	198	192
193	189	205	198	216	141	192	203	194	199	196
196	217	207	186	146	207	211	188	203	188	180

L'objectif est de réaliser deux types de filtrage sur quelques éléments de cette fenêtre.

1 - Filtrage Passe-Bas

On considère un filtre passe-bas binomial (3 × 3), dont le maque de convolution est le suivant :

$$\frac{1}{16} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

L'application de ce maque sur la zone (20 × 11) considérée délivre le résultat partiel suivant :

111	149	149	150	149	146	148	150	150	149	113
149	199	198	199	199	198	201	203	201	197	149
149	200	199	198	198	199	202	204	200	188	135
148	200		198	201	203	200	196	185	161	105
146	197	199	203	206	205	194	175	151	121	75
149	197	199	203	201		169	142	112	96	75
148	197	196	193	181	157	129	108	102		113
145	195	189	170	146	117	98		140	175	146
146	185	163	131	105	96	115	155	187	200	149
134	154	121	98		126	169	197	202	200	147
99	111	95	109	143	178	201	204	200	196	143
68		117	158	189	200	201	200	199	193	138
78	130	169	193	199	196	196		199	198	145
123	185	200	199	196	195	196	197	200	198	143
150	204	200	194	196	197	197	196	194	187	134
148	197	194		194	196	196	191		184	141
142	192	195	195	196	194	189	183	184	193	149
142	193	197	201	197	186	179	182	191	198	148
146	197	200	198	189		184	191	195	195	144
112	153	151	141	133	139	147	148	147	144	105

Renseignez les zones grisées afin d'obtenir la matrice I_S résultat (arrondir à la partie entière de l'élément calculé).

2 - Filtrage Rehausseur de contraste

On souhaite à présent améliorer la qualité de cette portion d'image en accentuant le contraste sur les bords des objets. Pour cela, on utilise le filtre *enhancement* dont le masque de convolution est le suivant :

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

L'application de ce masque sur la zone considérée donne le résultat partiel suivant :

586	373	408	410	410	343	382	405	417	399	589
383	291	117	252	178	238	209	188	220	126	492
412	100	265	121	283	71	277	223	226	226	395
389	309	201	200		257	155	183	325	160	210
321	186	127	221	207	301	237	208	123	162	78
451	170	244	275	216	212	189	198	90		9
441	207	133	185	236	168	161	40	-3	-273	705
290	202	271	262	189	127		-167		497	424
437	236	304	18	40	25	-263	403	268	228	332
531	186		132		-127	536	253	223	183	367
338	39		-265	210	387	224	173	152	247	460
145	-11	-262	510	292	257	149	262		217	130
-66		520	151	194	147		238	141	274	487
392	384	212	182	228	160	252	158	211	200	403
404	262	202	151		181	235	150	295	270	142
465	165	181	148	214	207	128	301			533
285	195	221	200	157	195	316	141	121	352	373
426	172	210	198	238	275		120	251	184	368
383	140	230	183	387	-105	251	270	167	219	409
570	493	427	379	121	537	468	323	445	358	516

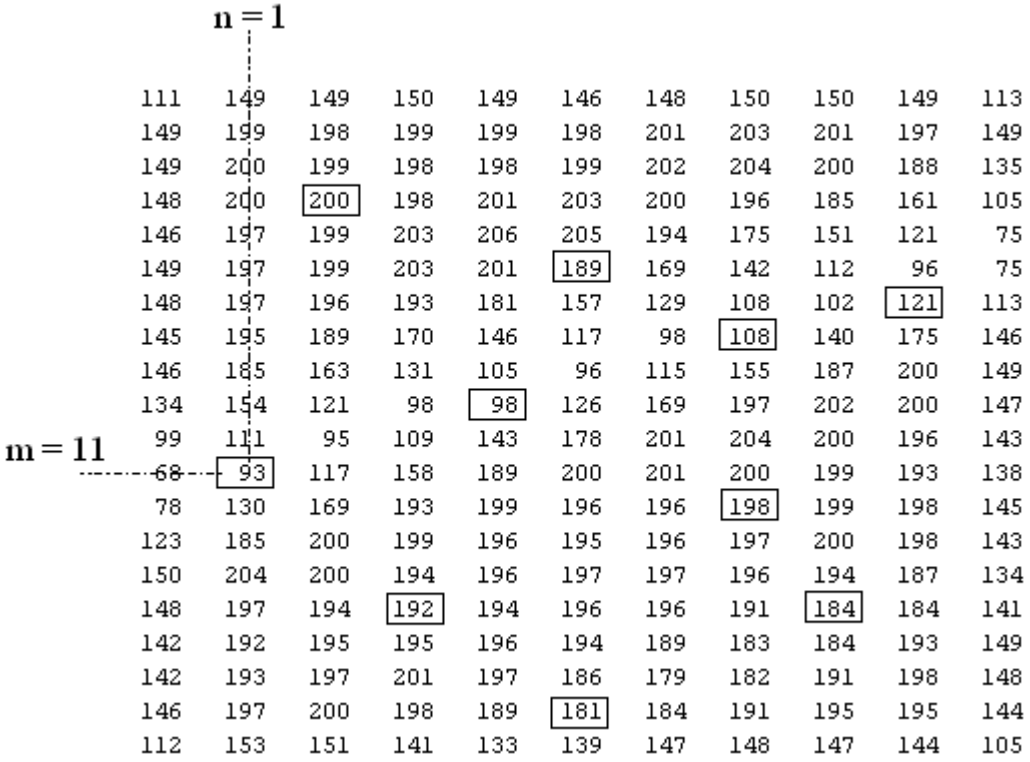
Comme pour le filtre passe-bas précédent, renseignez les zones grisées afin d'obtenir la matrice I_S résultat.

3 - Effets de bords

Un pixel situé sur le bord de l'image n'a pas de voisinage en dehors de l'image. On ne peut donc pas, à priori, calculer directement l'élément de bord obtenu après convolution. Proposez des méthodes pour traiter tout de même les bords.

Correction de l'exercice : Convolution

1 – Voici la matrice obtenue après filtrage passe-bas, avec le filtre binomial (3 × 3) :



Le calcul est présenté en détail pour l'élément $I_S(11, 1)$:

$$I_S(11,1) = \frac{1}{16} \cdot \sum_{i=-1}^{+1} \sum_{j=-1}^{+1} h(i,j) \cdot I_0(11-i, 1-j)$$

$$= \frac{1}{16} \cdot \sum_{i=-1}^{+1} h(i,-1) \cdot I_0(11-i, 1-(-1)) + h(i,0) \cdot I_0(11-i, 1-0) + h(i,1) \cdot I_0(11-i, 1-1)$$

$$= \frac{1}{16} \cdot \Big[h(-1,-1) \cdot I_0(11+1, 2) + h(-1,0) \cdot I_0(11+1, 1) + h(-1,1) \cdot I_0(11+1, 0)$$

$$+ h(0,-1) \cdot I_0(11, 2) + h(0,0) \cdot I_0(11, 1) + h(0,1) \cdot I_0(11, 0)$$

$$+ h(1,-1) \cdot I_0(12-1, 2) + h(1,0) \cdot I_0(12-1, 1) + h(1,1) \cdot I_0(12-1, 0) \Big]$$

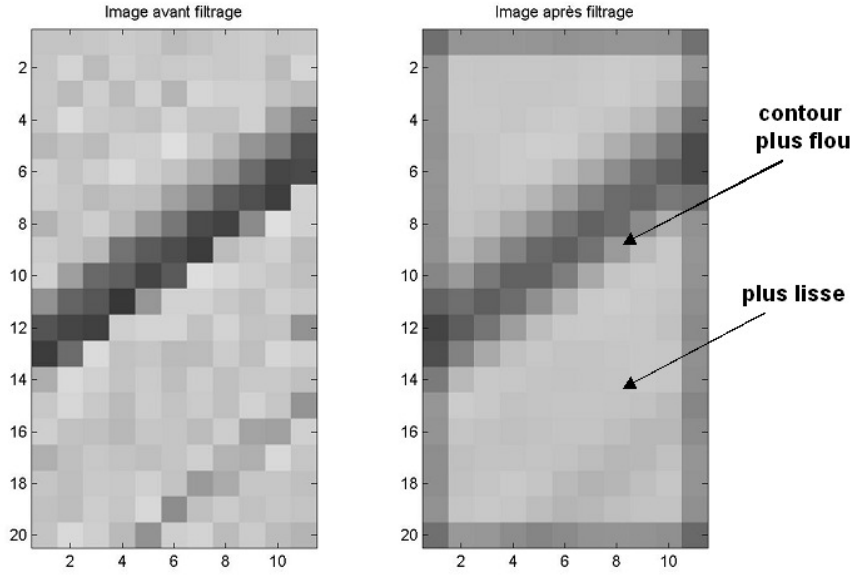
$$= \frac{1}{16} \cdot [\, 1\times219 + 2\times108 + 1\times60 + 2\times64 + 4\times69 + 2\times84 + 1\times87$$

$$+ 2\times100 + 1\times146 \,]$$

$$= 1500/16 = 93.75$$

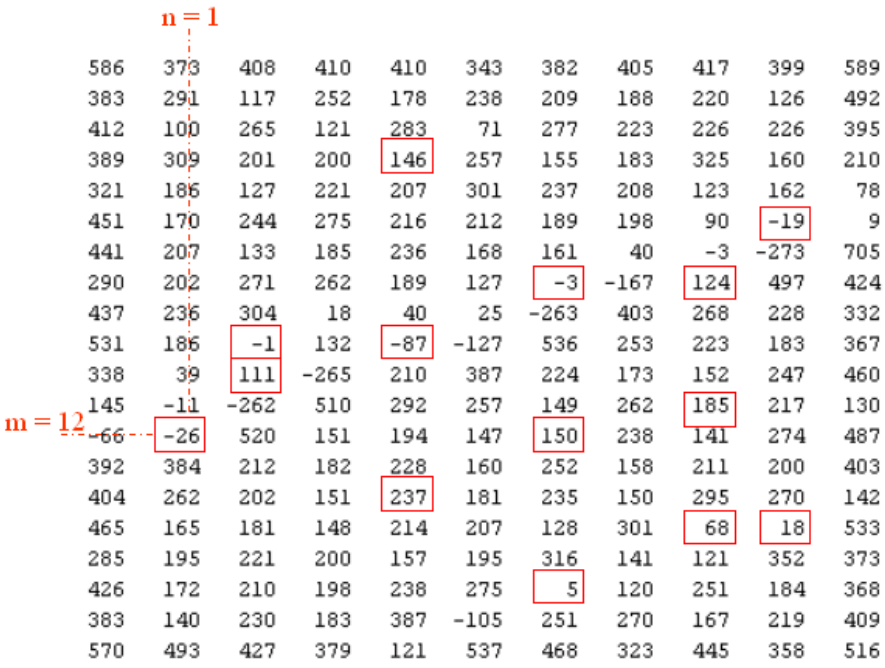
Donc : $I_S(11,1) = E[\, 93.75 \,] = 93$

Voici, sous Matlab, la visualisation de la matrice avant et après filtrage :



On voit très nettement l’effet de lissage du filtre passe-bas, engendré par la suppression des hautes et moyennes fréquences spatiales.

2 - Après application du filtre *enhancement* 3 × 3, la matrice obtenue est la suivante :



586	373	408	410	410	343	382	405	417	399	589
383	291	117	252	178	238	209	188	220	126	492
412	100	265	121	283	71	277	223	226	226	395
389	309	201	200	146	257	155	183	325	160	210
321	186	127	221	207	301	237	208	123	162	78
451	170	244	275	216	212	189	198	90	-19	9
441	207	133	185	236	168	161	40	-3	-273	705
290	202	271	262	189	127	-3	-167	124	497	424
437	236	304	18	40	25	-263	403	268	228	332
531	186	-1	132	-87	-127	536	253	223	183	367
338	39	111	-265	210	387	224	173	152	247	460
145	-11	-262	510	292	257	149	262	185	217	130
-66	-26	520	151	194	147	150	238	141	274	487
392	384	212	182	228	160	252	158	211	200	403
404	262	202	151	237	181	235	150	295	270	142
465	165	181	148	214	207	128	301	68	18	533
285	195	221	200	157	195	316	141	121	352	373
426	172	210	198	238	275	5	120	251	184	368
383	140	230	183	387	-105	251	270	167	219	409
570	493	427	379	121	537	468	323	445	358	516

Voici, à titre d'exemple, le détail du calcul pour la luminance du pixel d'affixe (12, 1) :

$$Is(12,1)=\sum_{i=-1}^{+1}\sum_{j=-1}^{+1} h(i,j).I_0(12-i,1-j)$$

$$=\sum_{i=-1}^{+1} h(i,-1).I_0(12-i,1-(-1))+h(i,0).I_0(12-i,1-0)+h(i,1).I_0(12-i,1-1)$$

$$=\Big[h(-1,-1).I_0(12+1,2)+h(-1,0).I_0(12+1,1)+h(-1,1).I_0(12+1,0)$$

$$+h(0,-1).I_0(12,2)+h(0,0).I_0(12,1)+h(0,1).I_0(12,0)$$

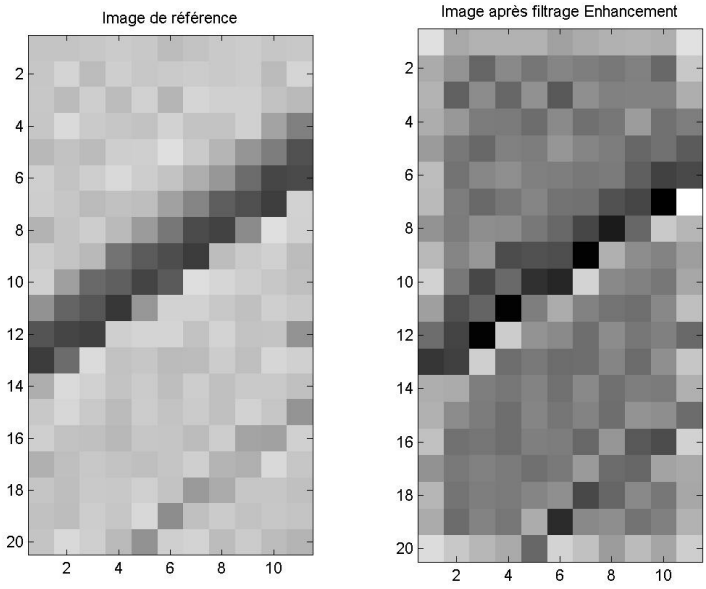
$$+h(1,-1).I_0(12-1,2)+h(1,0).I_0(12-1,1)+h(1,1).I_0(12-1,0)\Big]$$

$$= 0\times209 + (-1)\times218 + 0\times174 + (-1)\times219 + 5\times108 + (-1)\times60 + 0\times64 + (-1)\times69 + 0\times84$$

$$= -26$$

Notons que dans ce cas, les valeurs de luminance peuvent être négatives ou supérieures à 255. Pour l'affichage, il faut donc effectuer une remise à l'échelle des valeurs de luminance(*rescaling*).

Voici la visualisation de la matrice avant et après filtrage :



En fait, le filtre *enhancement* utilisé ici est conçu en réalisant la somme d'un filtre identité et d'un filtre Laplacien (utile à la détection de contours) :

$$\underset{\text{Identité}}{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}} + \underset{\text{Laplacien}}{\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}} \equiv \underset{\text{Enhancement}}{\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}}$$

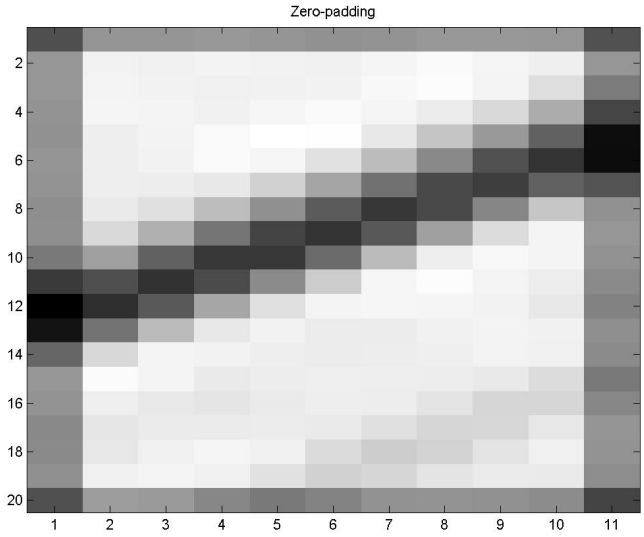
Le filtre *enhancement* est donc un filtre qui permet d'accentuer les contours (du tangon et du filin) dans l'image.

3 – Plusieurs méthodes existent pour traiter les bords d'une image : zero-padding, duplication, symétrie, ...
Pour plus d'informations sur ces méthodes, reportez vous à la correction de l'exercice du chapitre 3 : « Filtrage ».

Ici, nous présentons simplement quelques résultats obtenus, pour le filtrage passe-bas, avec différentes méthodes pour traiter les bords :

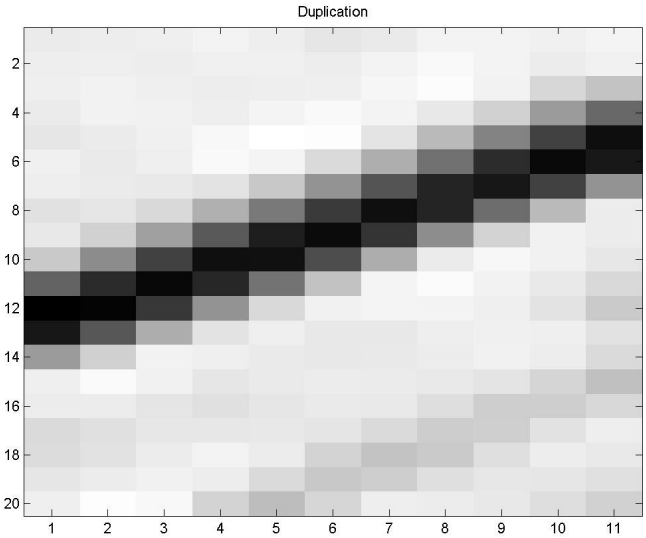
- Zero-padding :

Il s'agit du cas le plus simple : le voisinage en dehors de l'image est considéré comme un ensemble de pixels à valeurs nulles. On a donc l'apparition de bandes noires sur les bords :



- Duplication :

Le voisinage en dehors de l'image prend la valeur du pixel de l'image le plus proche (utilisation du paramètre 'replicate' dans la fonction ***imfilter*** sous Matlab). Il n'y a pas cette fois de phénomène de bandes noires :



Chapitre 3

Fondamentaux du traitement d'image

Filtrage Linéaire

Exemples de noyaux de filtres typiques

Support de taille (3×3) : le plus répandu et le plus simple

❶

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad h = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \qquad h = \begin{bmatrix} 1 & -3 & 1 \\ -3 & 9 & -3 \\ 1 & -3 & 1 \end{bmatrix}$$

Moyenneur — Binomial-gaussien — Rehausseur de contraste

❷

$$h = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad h = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \qquad h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Sobel horizontal — Gradient oblique — Laplacien

Filtres ❶ : $H(0,0) = \sum_i \sum_j h(i,j) = 1$

Filtres ❷ : $H(0,0) = \sum_i \sum_j h(i,j) = 1$

L'étude du noyau de convolution h permet de connaître la nature et les effets du filtre mis en oeuvre. Par exemple :

- si tous les coefficients du noyau sont positifs : le filtre est passe-bas et réalise une moyenne pondérée. Parfois, pour des pixels codés sur 8 bits, si le résultat de la somme pondérée est supérieure à 255, on le ramène par seuillage à 255. Ce type de filtrage a pour effet de lisser l'image afin de réduire, par exemple, le bruit ;
- si le noyau contient des coefficients positifs et négatifs, une différenciation est faite partiellement ou totalement. Le filtre correspondant a en partie ou totalement un comportement de type passe-haut. Une mise en évidence des fronts et des textures est alors obtenue ;

Des exemples de noyaux de filtre typiques sont ici présentés :

- des filtres-passe-bas : un moyenneur qui lisse l'image, le binomial-gaussien lisse aussi l'image de façon moins marquée mais plus régulière ;
- des filtres passe-haut : un rehausseur de contraste, des filtres différenciateurs orientés (Sobel horizontal, gradient oblique) et un différentiateur non orienté (laplacien).

Exemples d'application :

À titre d'exemple, un filtre passe-bas (3 × 3) « Binomial gaussien » et un filtre passe-haut (3 × 3) « Gradient oblique » sont appliqués à une partie de l'image *LENA* :

Après filtrage passe-bas par un filtre binomial-gaussien, les contours et les textures sont lissés. On observe ce phénomène notamment sur les zones d'ombre de l'épaule et du visage :

Après filtrage passe-haut par un filtre gradient oblique, les contours diagonaux de l'image sont mis en évidence :

Objectifs du filtrage

- Réduction du bruit
- Filtrage passe-bas avant sous-échantillonnage
- Amélioration du contraste

Filtres communs:

"Passe bas" (avec un gain DC unitaire)

$$H_1 = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad H_2 = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad H_3 = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Très bas — Bas — Moyen

(moyennage spatial)

"Passe haut" (avec un gain DC unitaire)

$$H'_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad H'_2 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix} \qquad H'_3 = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

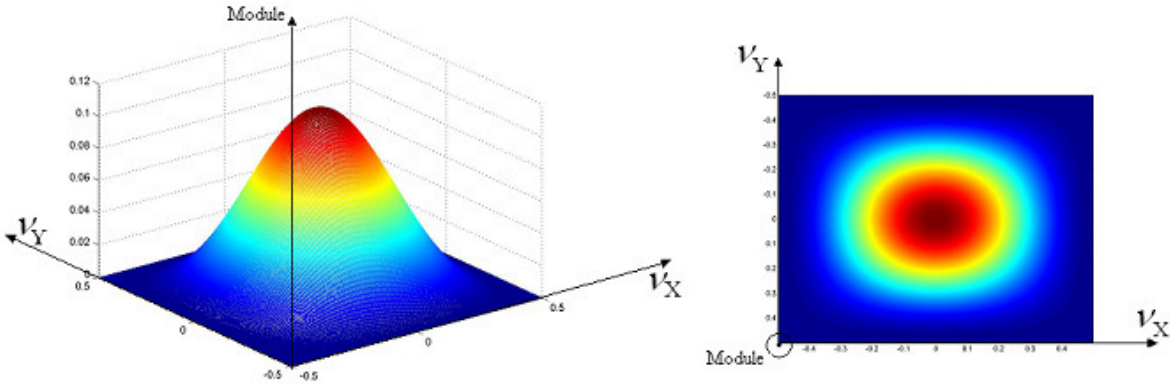
Le filtrage spatial peut donc être divisé en 2 catégories :

- **Le filtrage passe-bas qui atténue les variations de luminance**. Il a donc pour effet de lisser le contenu de l'image et de limiter les variations brusques d'intensité. On l'utilise typiquement afin de réduire les effets du bruit, et de supprimer le contenu haute fréquence (détails de l'image) avant un sous-échantillonnage (afin de limiter le risque d'*aliasing* ou d'apparition de fréquences parasites cf. chapitre 1) ;

- **Le filtrage passe-haut met en évidence les variations de luminance qui caractérisent traditionnellement les contours** des objets ou de la texture d'une image. Un filtre passe-haut avec un gain unitaire permet par exemple de rehausser le contraste de l'image originale.

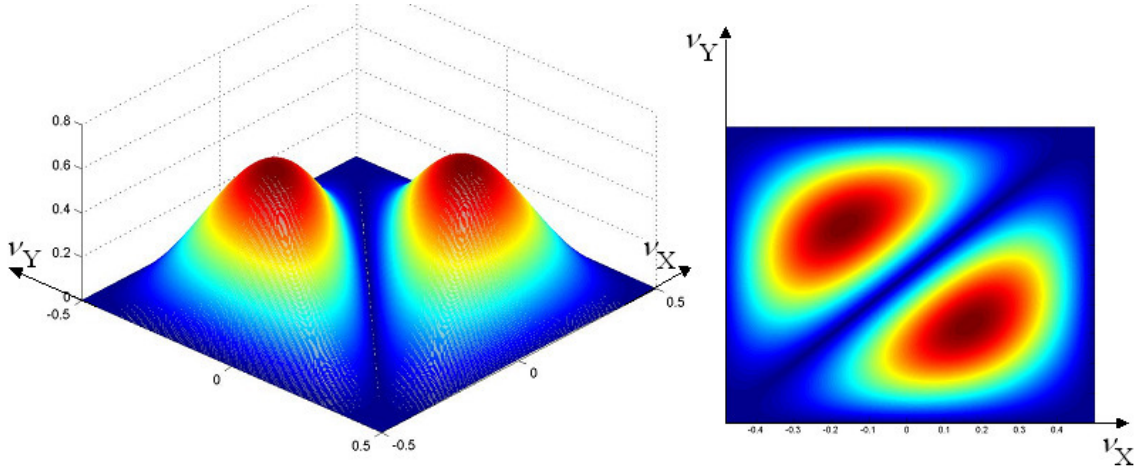
Il s'agit ici de présenter simplement quelques exemples de filtres typiques. Un utilisateur averti a la possibilité de choisir, voire de créer, le filtre correspondant à son besoin.

La figure ci-dessous présente le module du spectre obtenu par transformée de Fourier 2-D du filtre binomial gaussien H_3 (image de gauche) :



Le spectre bidimensionnel d'une image ou d'un filtre, représente une évolution du module de la transformée de Fourier selon les fréquences spatiales horizontales : « ν_X » et verticales : « ν_Y ». Usuellement, ce spectre est représenté par une vue de dessus, on parle alors de représentation dans le « plan de Fourier » (image de droite).
Sur le spectre du filtre binomial gaussien, le gabarit est celui d'un filtre passe-bas pour les fréquences horizontales et verticales. Sur l'image filtrée, les variations de luminance selon les axes horizontaux et verticaux sont donc diminuées : l'image est « lissée ».

Voici le spectre du filtre passe-haut « gradient oblique » utilisé précédemment sur une zone de l'image *LENA* :



Le filtre laisse passer les hautes fréquences diagonales. Sur l'image filtrée, les variations de luminances selon les directions diagonales sont accentuées. Les contours et les détails diagonaux sont mis en évidence.

Les concepts liés à la transformée de Fourier 2-D d'une image ou d'un filtre et aux fréquences spatiales sont plus amplement développés dans l'exercice « FFT » de ce même chapitre.

Exemples de filtres passe-bas

- Trois exemples de noyaux de convolution de filtres passe-bas à appliquer à l'image Lena.

- Gains en continu unitaires

	0	
0	¼	¼
	¼	¼

Filtre 1

	0		
0	1/9	1/9	1/9
	1/9	1/9	1/9
	1/9	1/9	1/9

Filtre 2

	0		
	0	1/8	0
0	1/8	1/2	1/8
	0	1/8	0

Filtre 3

On se propose d'appliquer, à titre d'exemple, trois filtres passe-bas différents à l'image monochrome *Lena*. Ces trois filtres sont à gain unitaire. Pour chacun d'entre eux, l'origine h(0,0) du noyau est indiquée par les marqueurs « 0 » horizontaux et verticaux.

Remarque : les filtres « 1 » et « 2 » sont des filtres moyenneurs de tailles respectives 2×2 et 3×3. Après filtrage d'une image, chaque pixel a une valeur de luminance qui est la valeur moyenne des luminances de son voisinage (de taille 2×2 pour le filtre 1 et 3×3 pour le filtre 2) : l'image est donc « lissée ».

Image originale *Lena*

Filtres (1) (2) (3)

En haut l'image originale. En bas, de gauche à droite, les images filtrées respectivement par les filtres 1, 2 et 3. On parvient à distinguer sur les ombres du visage de *Lena* un effet de lissage après filtrage passe-bas, cependant les effets sont difficilement observables à l'œil nu. L'œil humain est en fait peu sensible aux hautes fréquences spatiales et il ne voit pas que la suppression de ces hautes fréquences a entraîné la perte des détails de l'image originale. Le filtrage passe bas n'entraîne donc ici que peu de pertes en qualité visuelle et permet d'effectuer un sous échantillonnage sans effet de repliement du spectre (*aliasing*).

Filtres 2-D séparables: convolution de 2 filtres 1-D

$$[\,h_{2\text{-}D}\,] \equiv [\,h_{1\text{-}D}^{(V)}\,] \otimes [\,h_{1\text{-}D}^{(H)}\,]$$

$I_s(m,n) = h(m,n) \otimes I_e(m,n)$: filtrage linéaire 2-D

$I_s(m,n) = h_V(m) \otimes [h_H(n) \otimes I_e(m,n)]$: décomposition en deux filtres 1-D

$I_s(m,n) = h_V(m) \otimes I'(m,n) = h_H(n) \otimes I''(m,n)$: décomposition commutative

Avec:

$I'(m,n) = h_H(n) \otimes I_e(m,n)$ filtrage 1-D des lignes de l'image

$I''(m,n) = h_V(m,n) \otimes I_e(m,n)$ filtrage 1-D des colonnes de l'image

• ***Contraintes*** *: les poids des colonnes et lignes de* h_{2D} *doivent être proportionnels*

Exemple 1

$$[a\ b\ c] \otimes \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} a\alpha & b\alpha & c\alpha \\ a\beta & b\beta & c\beta \\ a\gamma & b\gamma & c\gamma \end{bmatrix}$$

Exemple 2

$$[a\ b\ a] \otimes \begin{bmatrix} a \\ b \\ a \end{bmatrix} = \begin{bmatrix} a^2 & ab & a^2 \\ ab & b^2 & ab \\ a^2 & ab & a^2 \end{bmatrix}$$

Des filtres 2D (bidimensionnels) sont dits séparables s'il est possible de décomposer le noyau $[h_{2D}]$ en deux filtres mono-dimensionnels appliqués successivement en horizontal puis en vertical (ou inversement) : $[h_{2D}] = [h_{1D}^{(V)}] \otimes [h_{1D}^{(H)}]$, où le symbole $\otimes$ désigne le produit de convolution.

Il est donc possible de traiter séparément les lignes et les colonnes de l'image originale. Le résultat final $I_S(m, n)$ reste bien évidemment identique quel que soit le premier filtre mono-dimensionnel appliqué. Pour qu'un filtre 2-D soit séparable, il faut que les coefficients de ses lignes et de ses colonnes soient proportionnels : mathématiquement c'est rarement le cas, cependant plusieurs filtres usuels sont séparables.

Exemples de filtres séparables

- Moyenneur

$$\frac{1}{3}\begin{bmatrix}1 & 1 & 1\end{bmatrix} \otimes \frac{1}{3}\begin{bmatrix}1\\1\\1\end{bmatrix} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$$

- Amélioration de contraste

$$\begin{bmatrix}-1 & 3 & -1\end{bmatrix} \otimes \begin{bmatrix}-1\\3\\-1\end{bmatrix} = \begin{bmatrix}1 & -3 & 1\\-3 & 9 & -3\\1 & -3 & 1\end{bmatrix}$$

- Binomial gaussien

$$\frac{1}{4}\begin{bmatrix}1 & 2 & 1\end{bmatrix} \otimes \frac{1}{4}\begin{bmatrix}1\\2\\1\end{bmatrix} = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$$

- Sobel horizontal

$$\begin{bmatrix}-1 & 0 & 1\end{bmatrix} \otimes \begin{bmatrix}1\\2\\1\end{bmatrix} = \begin{bmatrix}-1 & 0 & 1\\-2 & 0 & 2\\-1 & 0 & 1\end{bmatrix}$$

- Avantage : moins complexe

  Filtre 2D : $M \times N$ multiplications et additions par pixel (MAP)

  Filtre 2D séparable : $M + N$ multiplications et accumulations par pixel

Des exemples de filtres séparables sont présentés ici avec leur décomposition en filtres 1D. Les filtres séparables sont intéressants car ils permettent de réduire de façon parfois importante le nombre d'opérations (produits et additions) et donc le temps d'exécution. D'une façon générale, si le filtre est de taille M×N, on a besoin de seulement (M+N) multiplications et (M+N-2) additions au lieu de M.N multiplications et M.N-1 additions pour un filtre 2-D non séparable. Souvent on préfère parler d'opérations de type multiplications avec accumulation par pixel traité (noté 'MAP'). On passe donc de M.N MAP pour un filtre non séparable à (M+N) MAP pour un filtre séparable.

Exercice Chapitre 3 – Transformée de Fourier rapide (FFT)

Dans cet exercice, vous allez observer concrètement, sur divers exemples d'images, la représentation des spectres en fréquences spatiales.
Avant de commencer, chargez et décompressez le fichier archive «***fft.zip***» qui contient les scripts nécessaires tout au long de cet exercice.

Transformée de Fourier Discrète

1 – Après avoir mis à jour la liste des chemins dans le path browser, ouvrez et analysez le script ***fft2d_sinus.m***. N'hésitez pas à consulter l'aide (***helpwin***) pour obtenir plus d'informations sur les différentes fonctions Matlab utilisées. Quel est le type du signal 2-D d'entrée ?

2 – Jouez sur la période en essayant les valeurs 4, 8 et 16 (orientation 0). Relevez, pour ces différentes périodes, l'amplitude et la fréquence normalisées des raies. Jouez également sur l'amplitude et la valeur moyenne du signal 2-D. Commentez les résultats obtenus. Essayez ensuite la valeur 17 pour la période. Commentez à nouveau.

3 – Pour augmenter la résolution fréquentielle, utilisez le script ***fft2d_resolution.m***. Faites varier maintenant l'orientation du signal 2-D : essayez pi/2, et pi/4 (période 16 et 16*sqrt(2)).

4 – Lancez les scripts ***fft2d_carre.m*** et ***fft2d_damier.m***. Jouez avec les paramètres et interprétez les résultats.

5 – À partir du script ***fft2d_sinus.m***, écrivez un script qui affiche une image naturelle et son module de FFT2D. Lancez ce script sur plusieurs images.

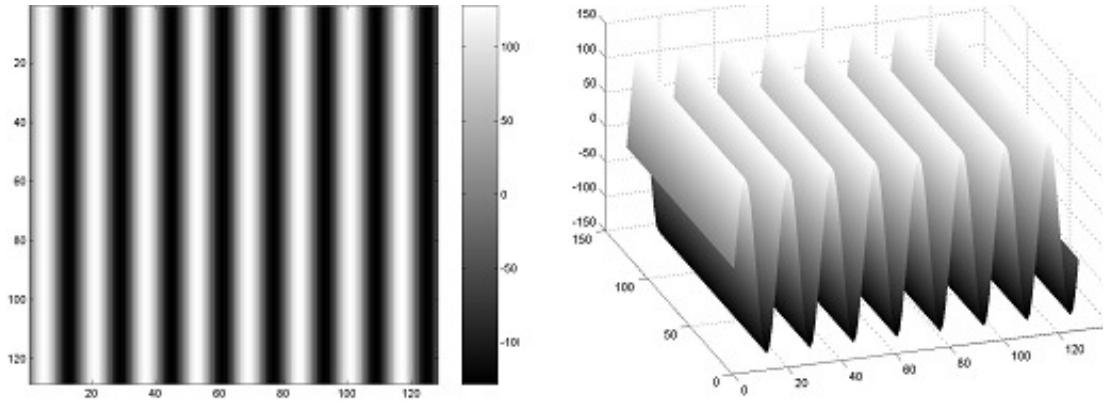
Correction de l'exercice : FFT

1 – Le fichier ***fft2d_sinus.m*** permet d'observer la transformée de Fourier discrète d'un signal 2-D sinusoïdal.

Dans un premier temps, on définit les caractéristiques du signal 2-D sinusoïdal que l'on veut utiliser : taille de l'image, période, orientation, amplitude, et valeur moyenne. Puis, la fonction ***generation_sinus***, qui vous est fournie ici, permet de générer un signal 2-D sinusoïdal à partir de ces caractéristiques.

```
taille = 128
periode = 16
orientation = 0
amplitude = 128
val_moy = 0
im1= generation_sinus(taille,periode,orientation,amplitude,val_moy);
```

L'image de la sinusoïde peut être ensuite visualisée de deux façons : en vue de dessus (vision 2-D avec la fonction ***imagesc***) et en perspective(vision 3-D avec la fonction ***surf***) :



On remarque que la direction horizontale de la sinusoïde entraîne l'apparition de bandes verticales sur l'image. Une fois que l'image de la sinusoïde est générée et stockée dans *im1*, on calcule sa transformée de Fourier discrète rapide (FFT) avec la fonction ***fft2*** de Matlab.

```
spectre1 = fft2(im1)/(taille*taille);
```

Puis, on utilise la fonction ***fftshift*** pour que la composante fréquentielle spatiale (0,0) soit déplacée au centre du support du spectre.

```
spectre1 = fftshift(spectre1);
```

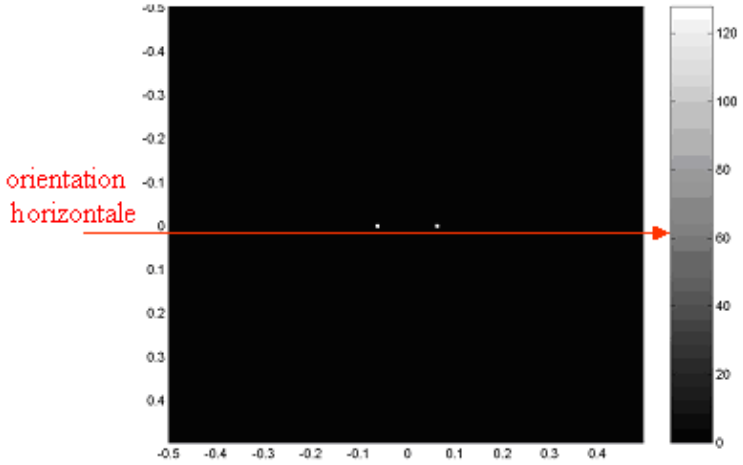
Enfin, les vecteurs normalisés représentent les fréquences spatiales horizontales et verticales. Ici, ces vecteurs possèdent la même taille et les mêmes composantes donc :

```
vt =(-taille/2:1:(taille/2-1))/taille;
```

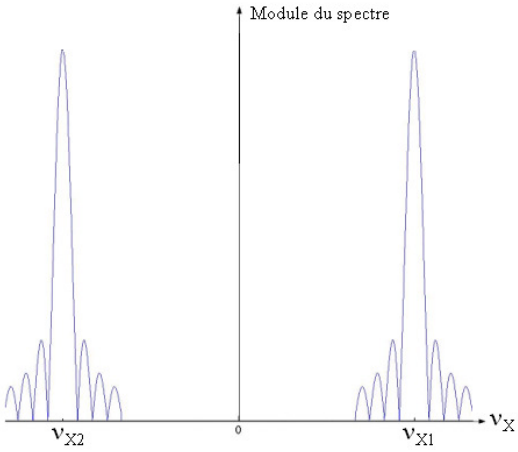
Le module du spectre peut s'afficher avec la commande :

```
imagesc(vt,vt,(abs(spectre1)));
```

L'image du module du spectre est la suivante (image aussi appelée « ***Plan de Fourier*** ») :



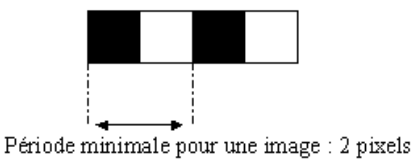
On aperçoit au centre deux points blancs. Dans la suite de l'exercice, on montrera qu'il ne s'agit pas de deux impulsions, mais des deux lobes principaux de sinus cardinaux, obtenus par transformation de Fourier rapide (FFT, pour *Fast Fourier Transform*) de la **sinusoïde sur un support fini** :



Enfin, on note que l'orientation définie par ces deux points est horizontale (elle est donc orthogonale aux orientations verticales des bandes que l'on observait sur l'image de la sinusoïde).

Remarque :

La période pour un signal d'image correspond au nombre de pixels minimal qui sépare deux motifs identiques dans l'image. La période minimale pour un motif dans une image est donc de deux pixels :



La fréquence spatiale normalisée maximale ν_{max} est donc égale à :

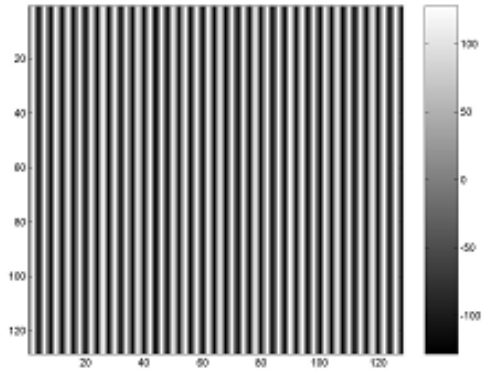
$$\nu_{max} = \frac{1}{\text{période minimale}} = \frac{1}{2}$$

En traitement d'images numériques, le module du spectre d'une image peut être représenté selon les **fréquences spatiales normalisées** qui varient donc sur l'intervalle [-0.5, 0.5].
Soit T la période en pixel, on définit alors la fréquence normalisée ν_{norm} correspondante par : $\nu_{norm} = 1/T$.

2 –

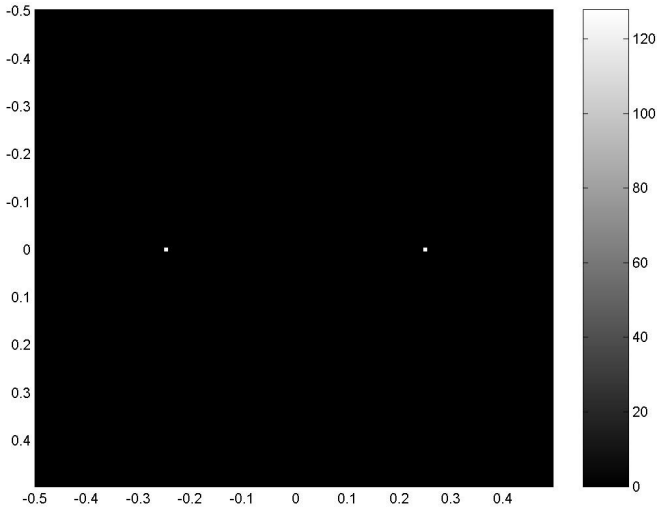
- Modification de la période :

En diminuant la période spatiale de la sinusoïde (valeur 4), on obtient les images suivantes :



Sur l'image, on peut voir une raie tous les quatre pixels.

L'image du module du spectre est donnée ci-dessous :



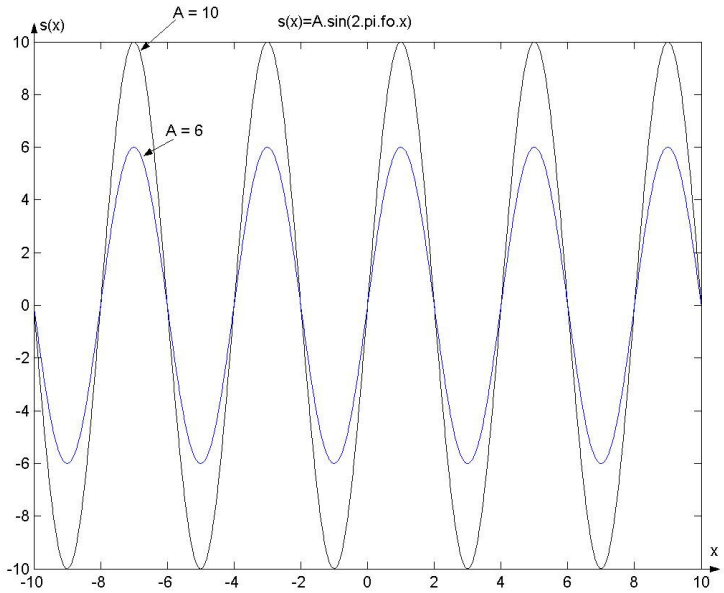
Les deux points sont aux fréquences normalisées :

$$\{\nu_{1X}= -0.25,\ \nu_{1Y}= 0\} \quad \text{et} \quad \{\nu_{2X}= 0.25,\ \nu_{2Y}= 0\}$$

Dans cette configuration particulière où l'orientation de la sinusoïde 2-D est à 0°, il est possible de représenter une ligne de ce signal 2-D par une sinusoïde 1-D qui se propage selon l'axe horizontal (Ox). Les fréquences spatiales horizontales ν_{1X}, et ν_{2X} sont donc égales, au signe près, à l'inverse de la période spatiale : $\nu_{1X}= -1/T$ et $\nu_{2X}= 1/T$.

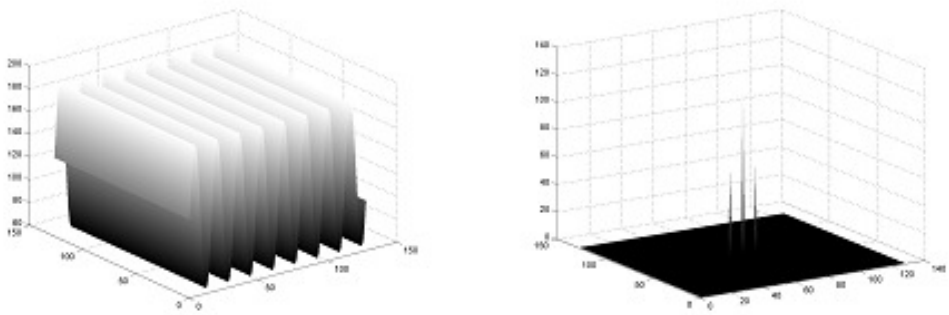
- Modification de l’amplitude

La sinusoïde évolue sur l’intervalle [-A, A], où A est l’amplitude de la sinusoïde. L’augmentation de l’amplitude A modifie l’amplitude des deux pics déjà observés.



- Modification de la valeur moyenne :

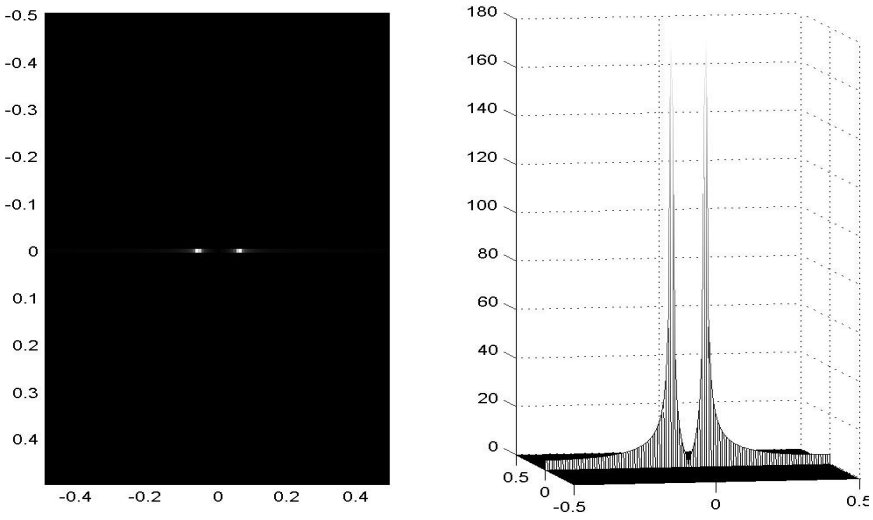
En augmentant la valeur moyenne V_M, on obtient les résultats suivants :



Le module du spectre, sur la figure de droite, présente en son centre une nouvelle composante en plus des deux composantes précédentes. Il s’agit en fait de la composante continue du spectre située aux fréquences spatiales $\{\nu_X = 0, \nu_Y = 0\}$ qui résulte de la valeur moyenne de la sinusoïde.

- Période T = 17 :

Pour une période T = 17, le module du spectre obtenu est le suivant :



Sur le plan de Fourier (à gauche), les deux points blancs sont « étalés ». Pour obtenir une meilleure visualisation, on affiche le module du spectre en vision 3-D (à droite). On observe alors que le module du spectre n'est plus composé de deux impulsions, et qu'il ressemble d'avantage à l'enveloppe d'un sinus cardinal.

Pour expliquer ce phénomène, on considère une sinusoïde 1-D, notée f(x) et définie par :

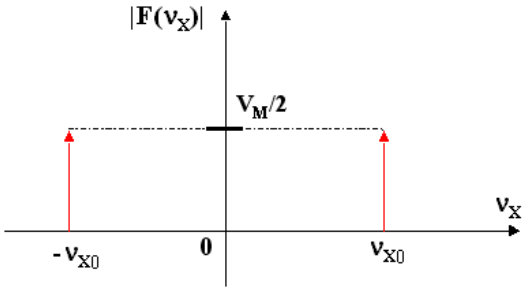
$$f(x)=V_M.\sin(2.\pi.f_0.x)$$

Le spectre $F(\nu_X)$ de cette sinusoïde est donc défini par :

$$F(\nu_X)=V_M.\left(\frac{1}{2j}.\delta(\nu_X-\nu_{X0})+\frac{j}{2}.\delta(\nu_X+\nu_{X0})\right)$$

où « j » désigne le nombre complexe tel que $j^2= -1$, et $\nu_{X0} = 1/T$.

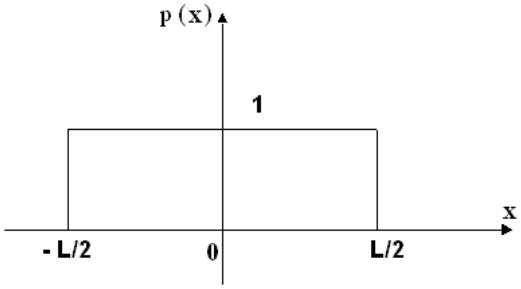
Le module de spectre $|F(\nu_X)|$ est représenté sur la figure ci-dessous :



Ce spectre est défini pour une sinusoïde de **support infini** (x variant de $-\infty$ à $+\infty$). En pratique une image n'est jamais représentée sur un support infini, on effectue donc un fenêtrage de la sinusoïde par une fonction porte notée p(x). La fonction p(x) est définie par :

$$p(x) = \begin{cases} 1 \text{ si } x\in[-L/2;L/2] \\ 0 \text{ sinon} \end{cases}$$

Représentation de p(x) :

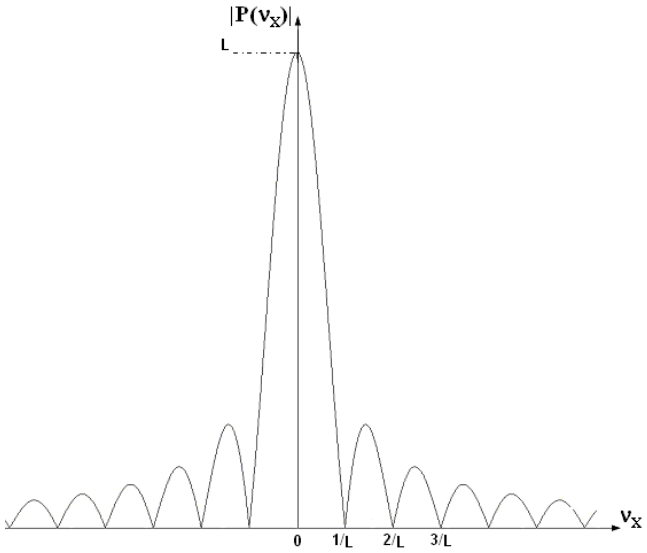


Le spectre $P(\nu_X)$ de cette fonction porte est défini par :

$$P(\nu_X)=L.sinc(L.\pi.\nu_X)$$

Où « sinc » exprime un sinus cardinal : sinc(x) = sin(x)/x.
Dans le script ***fft2d_sinus.m***, la taille de l'image est de 128 x 128 pixels. La sinusoïde se déplace selon (Ox), on a donc L= 128.

Le module du spectre $|P(\nu_X)|$ est représenté sur la figure ci-dessous :



On remarque sur cette figure que la largeur d'un lobe secondaire du sinus cardinal vaut 1/L.

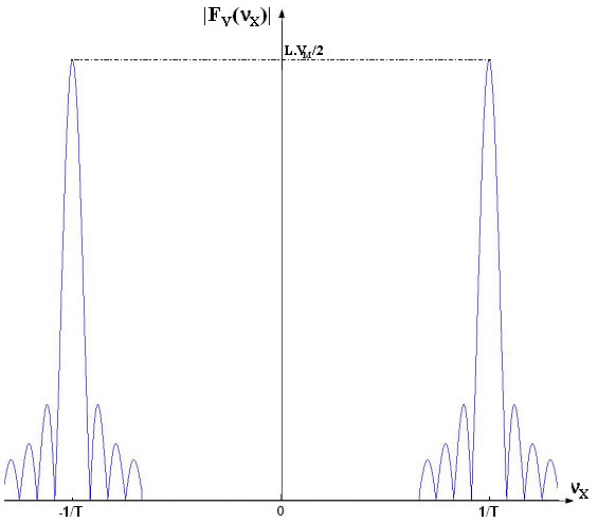
Ainsi, la sinusoïde $f_V(x)$ visualisée est, en réalité, une fonction de la forme :

$$f_V(x) = f(x).p(x)$$

Donc le spectre $F_V(\nu_X)$ est donnée par la relation :

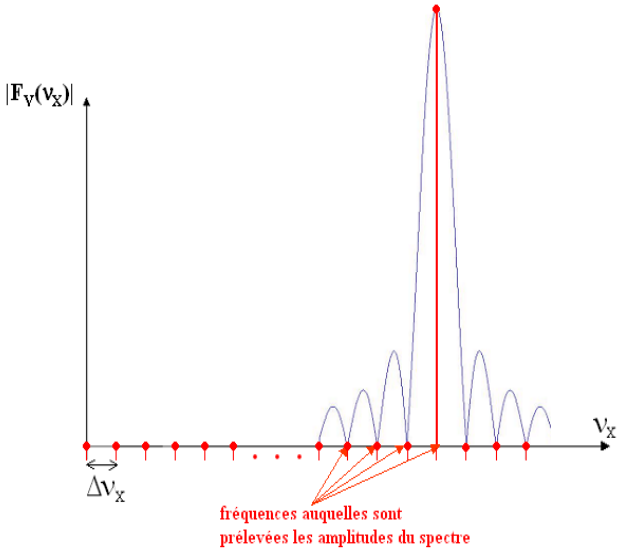
$$F_V(\nu_X) = F(\nu_X) \otimes P(\nu_X) = \frac{1}{2}.V_M.\left(\frac{1}{j}.P(\nu_X-\nu_{X0})+j.P(\nu_X+\nu_{X0})\right)$$

Le module du spectre $|F_V(\nu_X)|$ de la sinusoïde $f_V(x)$ de période T et d'amplitude V_M, sur un support fini de taille L, est représenté sur la figure ci-dessous :



En outre, pour représenter le spectre de cette sinusoïde, on calcule une Transformée de Fourier rapide (donc **numérique**). Le spectre est donc représenté par un nombre fini de ses points. Par défaut, avec la fonction ***fft2***, ce nombre de point est égal au nombre de pixels dans chacune des directions (horizontales et verticale).

Dans notre configuration, le nombre de pixels est le même selon les deux directions. Après calcul de la FFT, on a donc un module du spectre $|F_{Vs}|$ représenté par 128 points du module du spectre $|F_V|$ pour les fréquences spatiales horizontales. On a alors une résolution fréquentielle $\Delta\nu_X$ en horizontal définie par : $\Delta\nu_X = 1/128$.
On remarque que $1/L = \Delta\nu_X$: sur le spectre de la sinusoïde a support fini, la largeur d'un lobe secondaire du sinus cardinal est égale à la résolution fréquentielle en horizontal.
Ainsi, si la période T de la sinusoïde est inversement proportionnelle à la résolution fréquentielle horizontale (i.e. $1/T = k.\Delta\nu_X$, où « k » est un entier naturel non nul), le module du spectre $|F_{Vs}|$ ne représente que les valeurs maximales des sinus cardinaux du module du spectre $|F_V|$ (les lobes secondaires disparaissent) :

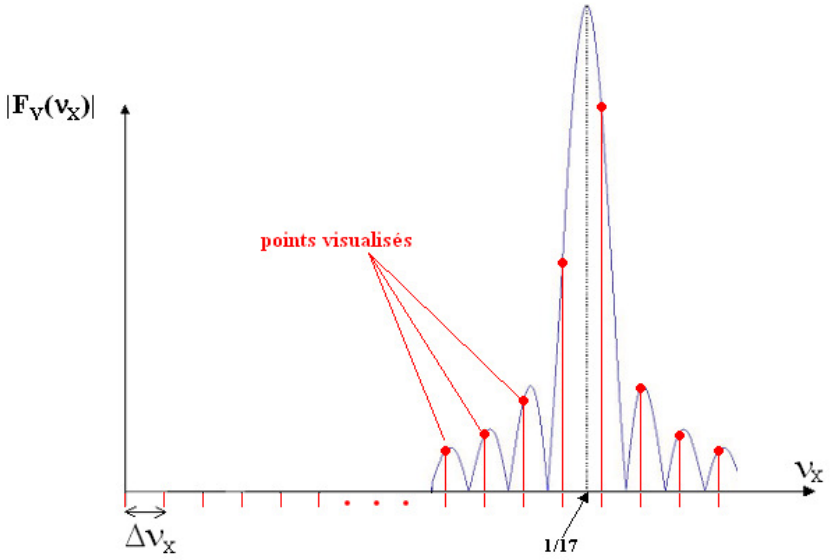


Ainsi pour les périodes :

- T = 4, $\frac{1}{4}=32\times\frac{1}{128}$;
- T = 8, $\frac{1}{8}=16\times\frac{1}{128}$;
- T = 16, $\frac{1}{16}=8\times\frac{1}{128}$;

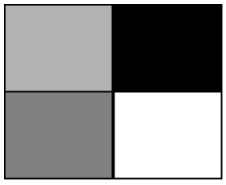
Pour ces trois périodes, la condition $1/T = k.\Delta\nu_X$ est vérifiée. On visualise donc uniquement la valeur maximale des sinus cardinaux. Les autres composantes du spectre étant nulles, le module du spectre ressemble à deux impulsions (cf. 1).

Pour une période T = 17, la condition $1/T = k.\Delta\nu_X$ n'est pas vérifiée. On visualise donc des échantillons des lobes secondaires des sinus cardinaux :

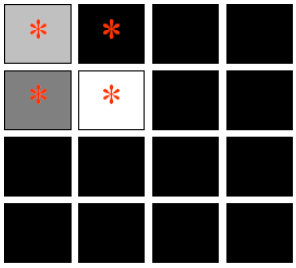


Pour pouvoir visualiser les lobes secondaires, quelle que soit la période, il faut augmenter la résolution fréquentielle pour satisfaire à la condition : $\Delta\nu_X < 1/L$.

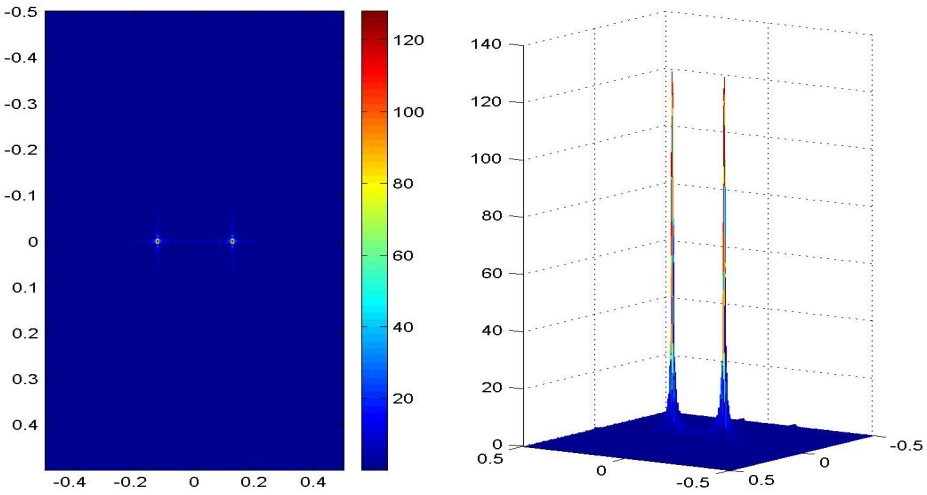
3 – Le script ***fft2d-resolution.m*** permet d'augmenter la **résolution fréquentielle** de l'image. On choisit pour une dimension (horizontale puis verticale, ou inversement) un nombre de points N_2 pour calculer la FFT. Ce nombre de points doit être supérieur au nombre de points N_1 de l'image sur la même dimension (horizontale ou verticale). Les points introduits ont la valeur zéro (*zero-padding*).
Prenons le cas simple d'une image monochrome de taille 2x2 (N_1=2) :



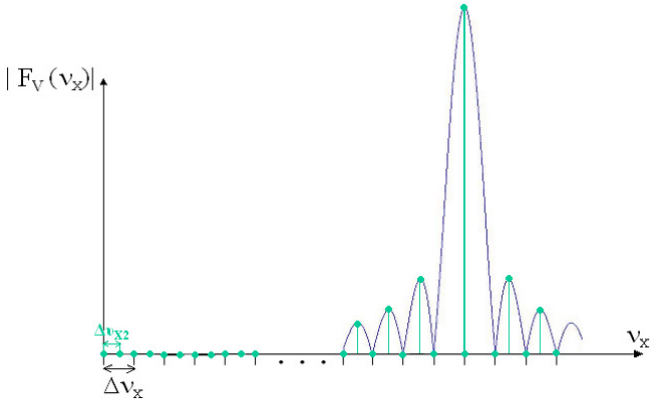
On choisit d'augmenter la résolution fréquentielle en calculant la FFT avec non plus 2, mais 4 points pour chaque dimension. La FFT sera donc calculée sur l'image suivante (N_2=4) :



Les quatre pixels de l'image de référence 2x2 sont marqués d'une étoile rouge. Les autres pixels noirs (valeur 0) ont été ajoutés pour obtenir une image de taille 4x4.
Pour calculer, sous Matlab, la FFT avec une plus grande résolution fréquentielle, il suffit de passer le nombre de points N_2 en paramètre de la fonction ***fft2*** : fft2(im1,N_2,N_2). Voici les résultats obtenus pour une période T = 8 et la sinusoïde 2-D :



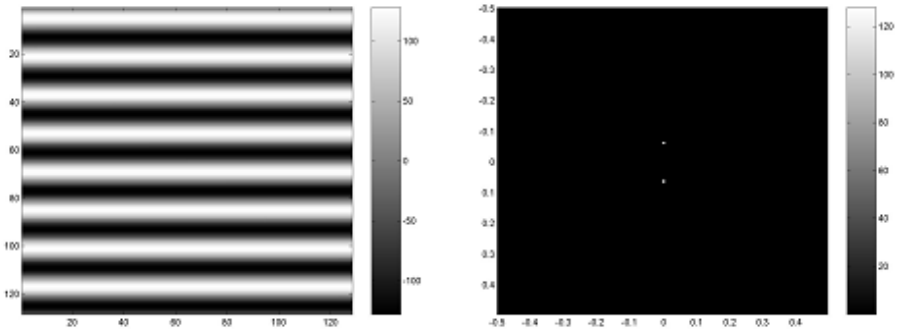
En augmentant la résolution fréquentielle, les lobes secondaires des sinus cardinaux du spectre apparaissent même pour les signaux dont la période vérifie 1/T = k.$\Delta\nu_X$. En effet, considérons, par exemple, une résolution fréquentielle doublée : $\Delta\nu_{X2}$= 2.$\Delta\nu_X$:



Les lobes secondaires du spectre sont maintenant représentés par des valeurs non nulles (valeurs maximales dans le cas présent).

Modification de l'orientation :

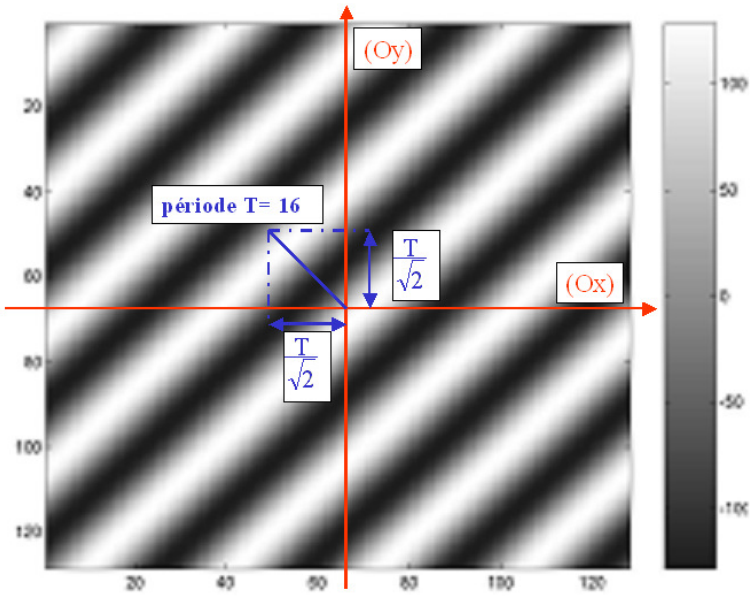
En prenant une orientation de pi/2 (période = 16 pixels), on obtient les résultats suivants :



Il était prévisible qu'en ajoutant à l'orientation de la sinusoïde un angle de pi/2 :

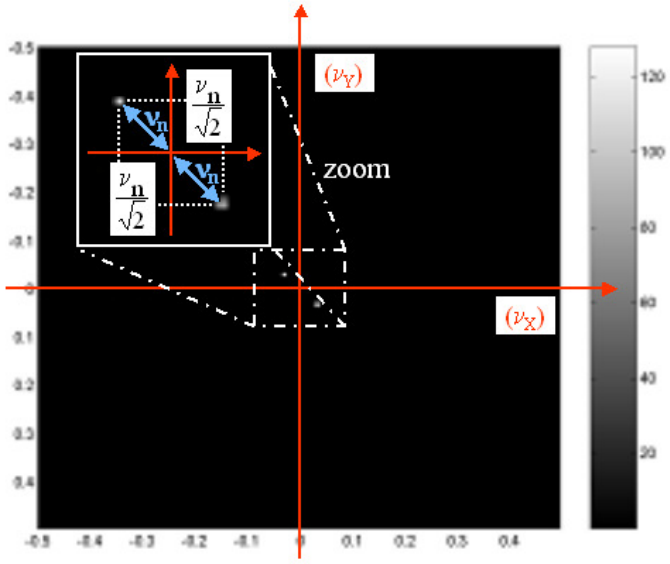
- la propagation soit perpendiculaire à celle de la sinusoïde de référence (les bandes sont maintenant horizontales dans l'image de gauche) ;
- la direction définie par les deux points du module du spectre soit perpendiculaire, à la fois, aux bandes de l'image qu'il représente et à la direction définie sur le spectre d'origine (image de droite).

De même en prenant une orientation de pi/4 (période = 16 pixels), l'orientation de la propagation de la sinusoïde a été modifiée d'un angle de –pi/4. On observe des structures périodiques horizontalement et verticalement. Ces structures contribuent à créer une structure périodique en diagonale de période T = 16. Les périodes horizontales et verticales sont donc égales à $\frac{T}{\sqrt{2}}$ pixels :

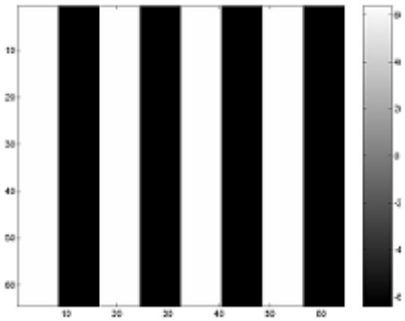


Sur l'image ci-dessous, le module du spectre est représenté par deux points situés respectivement aux coordonnées : $\left(-\frac{\nu_n}{\sqrt{2}};\frac{\nu_n}{\sqrt{2}}\right)$ et $\left(\frac{\nu_n}{\sqrt{2}};-\frac{\nu_n}{\sqrt{2}}\right)$, avec $\nu_n=\frac{1}{T}$.

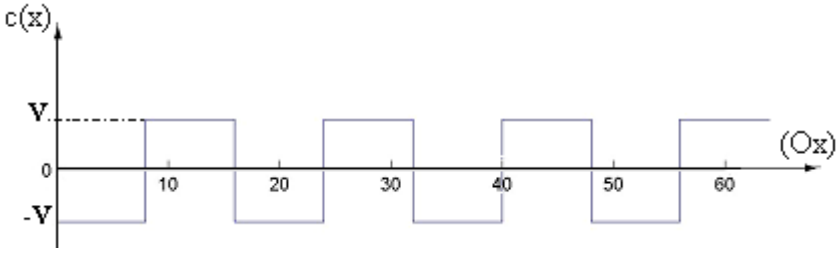
Ces points définissent une direction perpendiculaire aux bandes observées sur l'image précédente. L'analyse de Fourier 2-D fournit donc une indication sur l'orientation spatiale du signal.



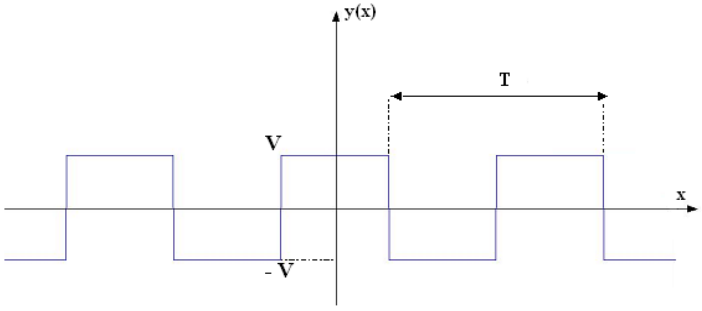
4 – Après avoir lancé le script ***ff2d_carre.m***, on obtient l'image suivante :



Le signal 2-D est un signal carré qui évolue selon l'axe horizontal (Ox). Il est possible de représenter une ligne de cette image par un signal carré 1-D selon l'axe (Ox) :

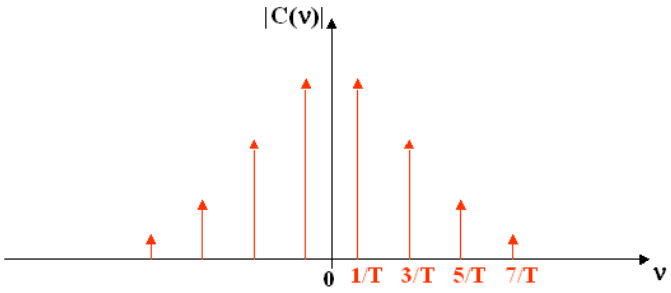


On sait que pour une onde carrée de la forme :



La transformée de Fourier est : $\mathbf{Y(\nu)=\sum_n V.sinc(n/2).\delta(\nu-n/T)}$. Si on note T_1 la période de c(x), on a donc : $c(x)=y(x+T_1/4)=y(x)\otimes\delta(x+T_1/4)$.

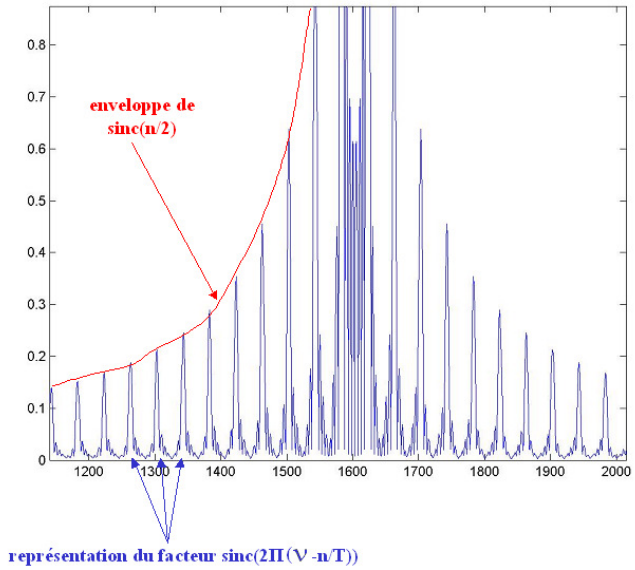
Ainsi : $\mathbf{C(\nu)=Y(\nu).exp(2j\pi\nu.T_1/4)}$, en module les spectre C(ν) et Y(ν) sont donc identiques et représentés par :



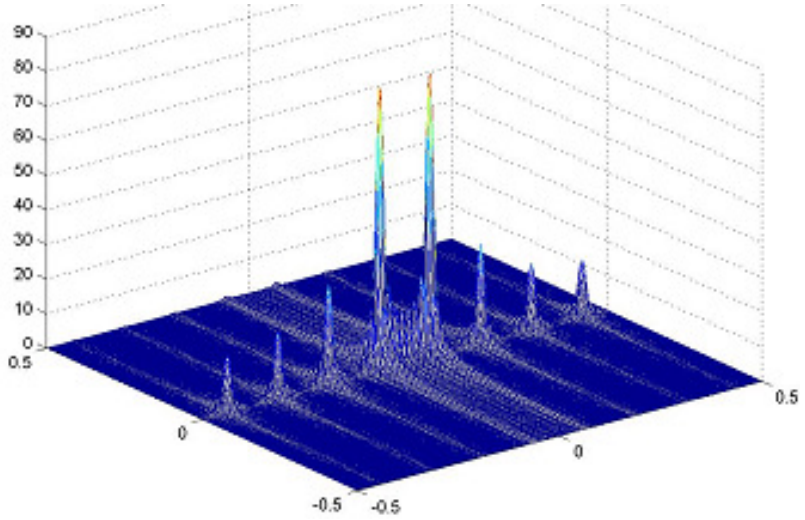
Ici, le signal est considéré infini. Ce n'est pas le cas dans la réalité pour une image, il faut donc - comme pour la sinusoïde - effectuer une multiplication par la fonction porte p(t) utilisée précédemment. On a donc : $\mathbf{|C(\nu)|=|\sum_n V.sinc(n/2).\delta(\nu-n/T)\otimes L.sinc(2\pi\nu)|}$

D'où : $\mathbf{|C(\nu)|=V.L.\sum_n |sinc(n/2).sinc(2\pi(\nu-n/T))|}$

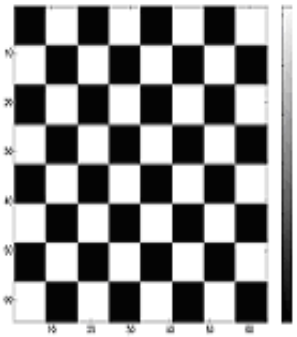
Le module du spectre C(ν) est donc obtenu par la multiplication de deux sinus cardinaux : un propre au support fini du signal, l'autre propre au signal carré proprement dit. Le module du spectre est donc obtenu en remplaçant, sur la figure précédente, les impulsions de Dirac par des sinus cardinaux. Notons que la valeur moyenne du signal est nulle, il n'y a donc pas de gain en continu. Ainsi, le module du spectre est de la forme suivante :

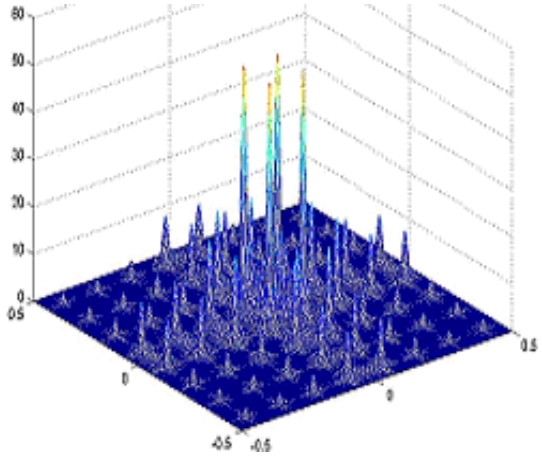


C’est le résultat qu’on observe sur la représentation du module du spectre du signal carré 2-D :



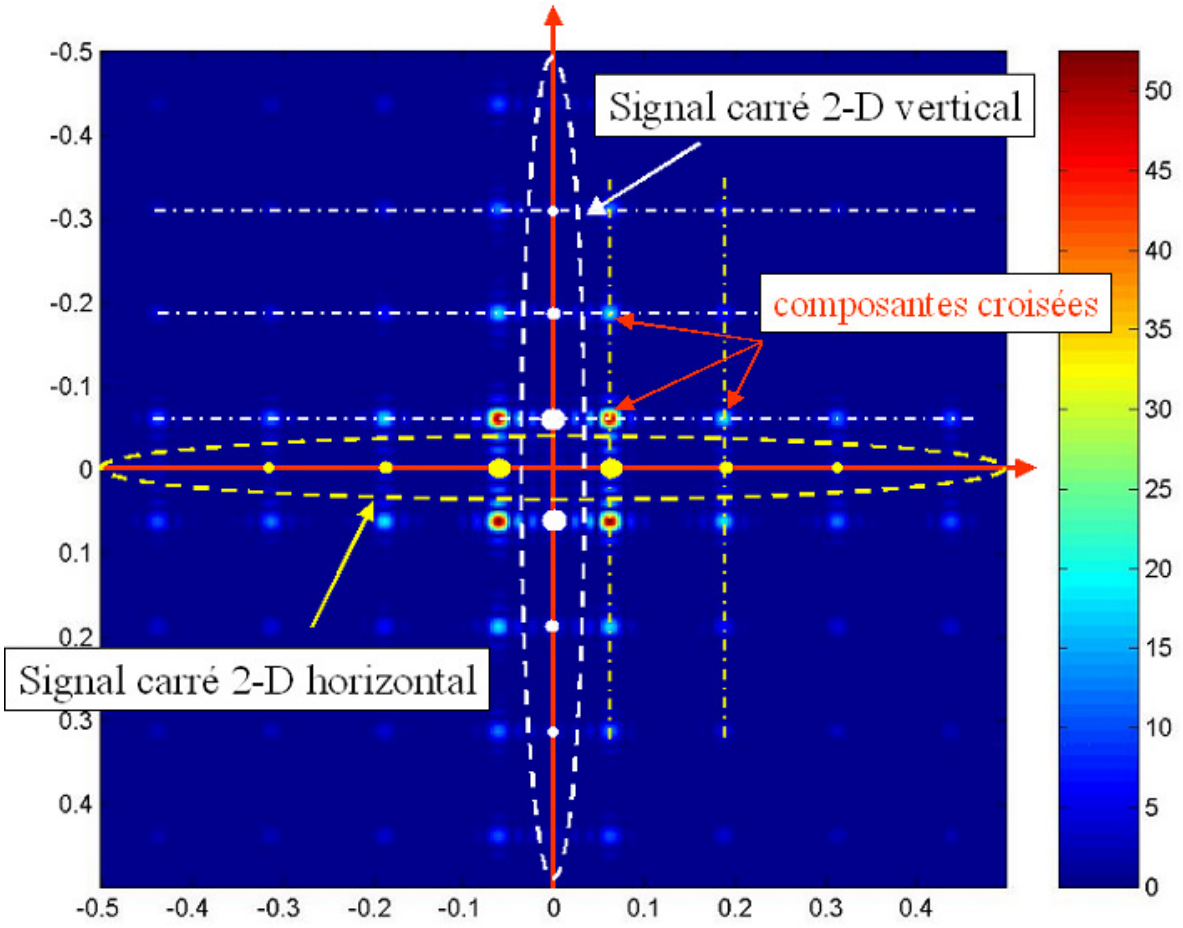
Après avoir lancé le script ***ff2d_damier.m***, on obtient l’image et le module du spectre associé suivants (période = 16) :





Le damier est obtenu en prenant, en entrée d’un opérateur « XOR », deux ondes carrées orthogonales entre elles (ici une onde verticale et une onde horizontale). Sur le module du spectre (à droite), on reconnaît alors des amplitudes croisées similaires à celles obtenues dans le cas d’une simple onde carrée.
Par ailleurs, sur l’image du damier (à gauche), on remarque que les structures sont à présent périodiques suivant les diagonales de l’image.

La figure ci-dessous présente le plan de Fourier pour l'image du damier (période = 16). On représente ici les modules des spectres des deux signaux carrés 2-D de base : un signal horizontal et l'autre vertical. Les composantes du module du spectre de l'image du damier résultent alors des croisements des composantes de ces deux signaux carrés de base.

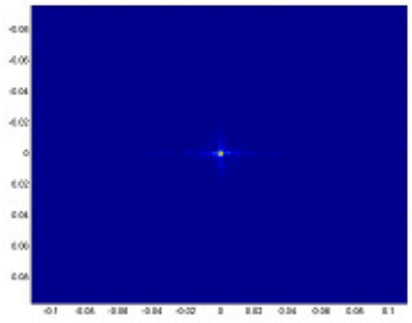


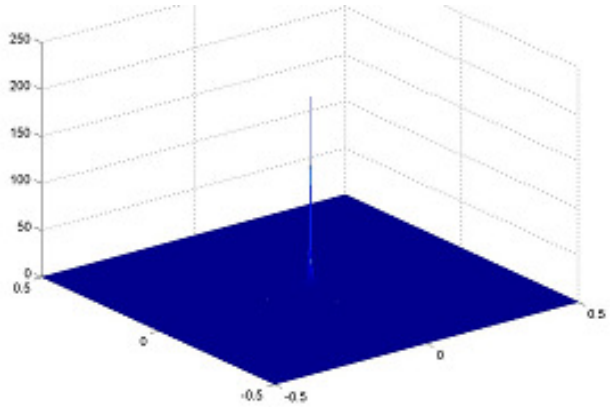
5 – Voici un exemple de script, inspiré de ***fft2d_sinus.m***, pour afficher le spectre de l'image *CLOWN_LUMI* :

```
im1=imread('CLOWN_LUMI.BMP') ;
% on affiche l'image en niveaux de gris
figure(1);
imagesc(im1);
map = 0:1/255:1;
map = [map',map',map']; % LUT pour afficher en niveau de gris
colormap(map);
% on calcule sa transformée de Fourier
[taille1, taille2]=size(im1(:,:,1)) ;
nb_point1 = 2*taille1;
nb_point2 = 2*taille2;
spectre1 = fft2(im1,nb_point1,nb_point2) /(taille1*taille2);
spectre1 = fftshift(spectre1);
%pour diminuer la composante continue
spectre1 = 2*spectre1;
spectre1(taille1/2+1,taille2/2+1)=spectre1(taille1/2+1,taille2/2+1)/2;
% on définit les vecteurs de fréquences normalisées
vt1=(-taille1/2:taille1/nb_point1:(taille1/2-taille1/nb_point1))/taille1;
vt2=(-taille2/2:taille2/nb_point2:(taille2/2-taille2/nb_point2))/taille2;
```

```
% on affiche le module de la FFT
figure(2);
imagesc(vt1,vt2,(abs(spectre1)));
%representation 3-D
figure(3)
[X,Y]=meshgrid(vt1,vt2);
mesh(X,Y,abs(spectre1));
```

On obtient le spectre suivant :

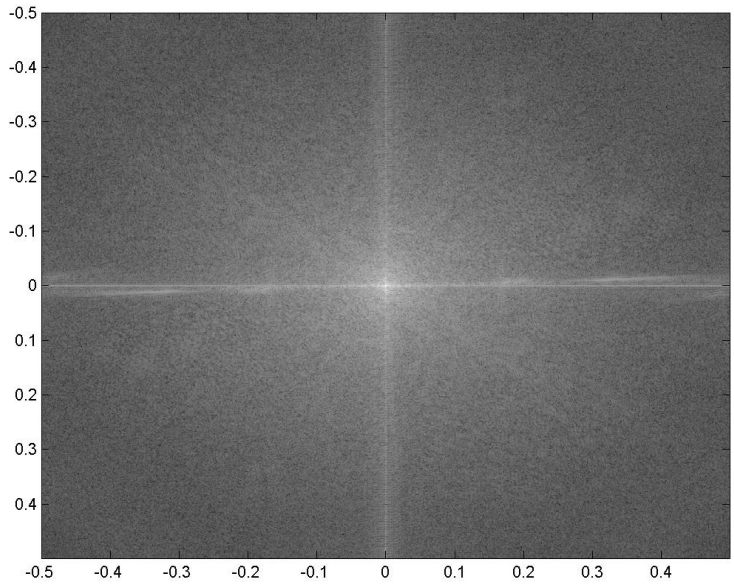




Comme pour la plupart des images naturelles, le spectre de l'image *CLOWN_LUMI* possède une forte composante continue, et très peu de composantes en moyennes et hautes fréquences spatiales.
Compte tenu de la dynamique du module du spectre (valeurs importantes en basses fréquences), on affichera le logarithme en base 10 de ce module :

```
% on affiche le module de la FFT
figure(2);
imagesc(vt1,vt2,(log10(abs(spectre1))));
colormap(map) ;
```

Voici le plan de Fourier obtenu alors :



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.

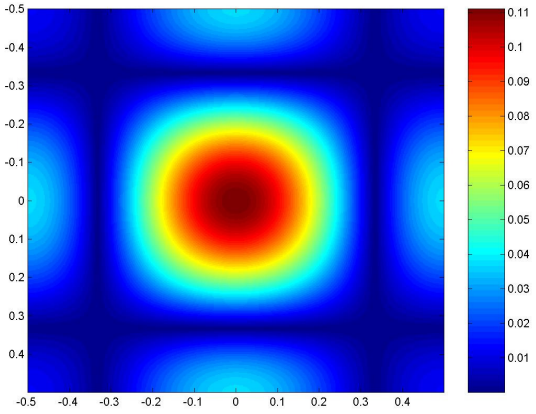
Correction de l'exercice : Filtrage linéaire

Filtrage linéaire dans le domaine fréquentiel et dans le domaine spatial

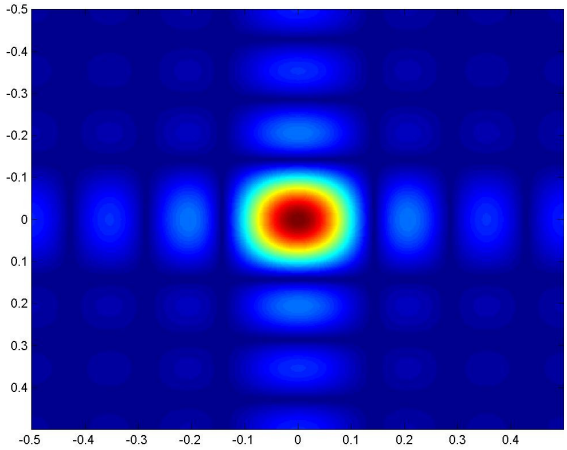
1 – La première partie du script ***filtrage_passebas.m*** définit le noyau du filtre passe bas qui sera utilisé. Dans le cas présent, il s'agit du noyau d'un filtre moyenneur qui donne à chaque pixel la valeur moyenne de son voisinage :

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

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

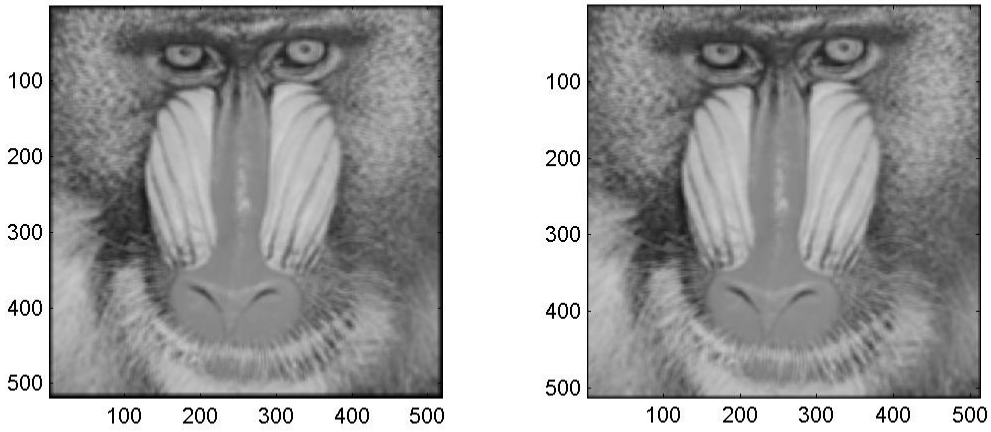


Sur l'image, on observe que le filtre moyenneur laisse passer les basses fréquences. Cependant, le gain remonte sur l'intervalle des moyennes et hautes fréquences spatiales normalisées. Le filtre passe-bas n'est donc pas très sélectif.
En augmentant la taille du support du filtre (support=7), on obtient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

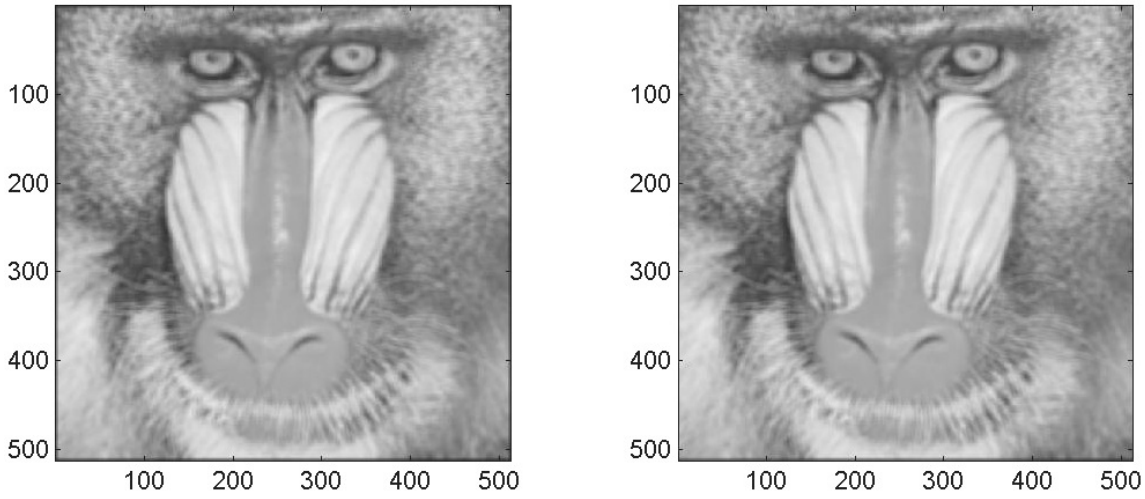
```
imf = conv2(im1,filtre); % (équivalent à imf=conv2(im1,filtre,'full');)
imf2 = conv2(im1,filtre,'same');
```

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.

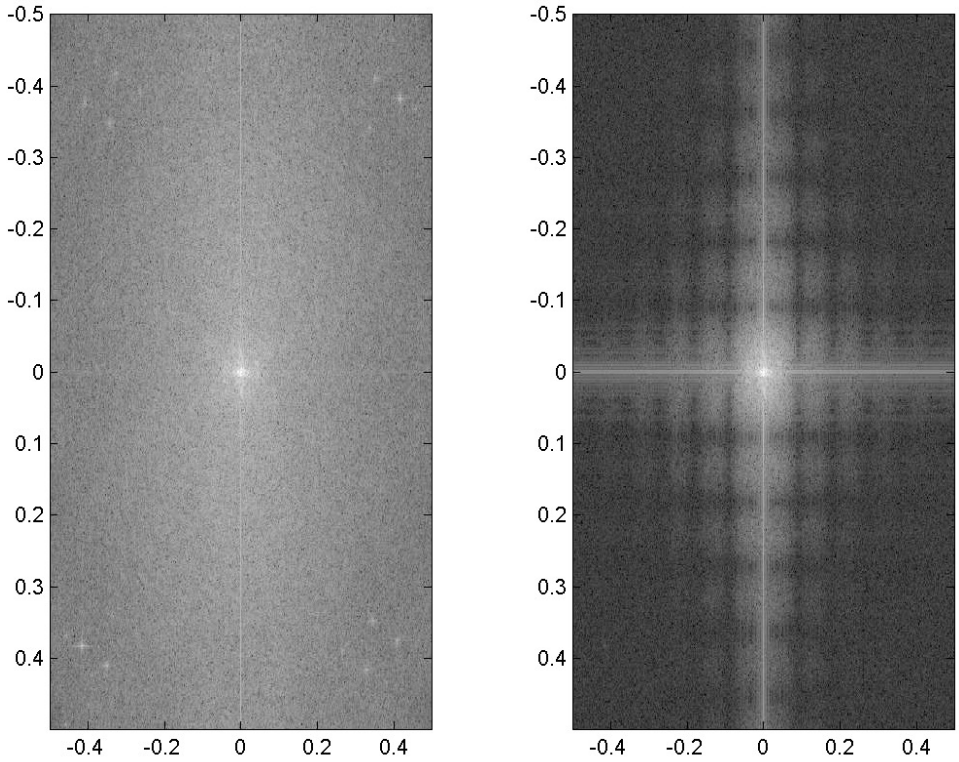
- Duplication :

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 :



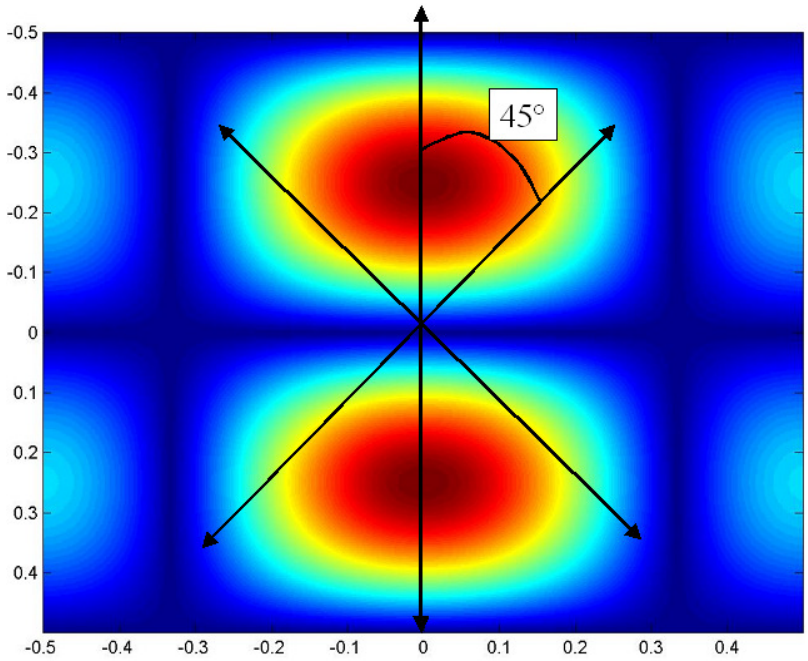
L'image à gauche représente le module du spectre de *MANDRILL_LUMI.BMP* avant filtrage. On aperçoit une zone claire au centre du spectre, qui correspond à la composante continue de l'image. Le nuage de points autour de cette composante continue correspond à quelques composantes moyennes et hautes fréquences 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).

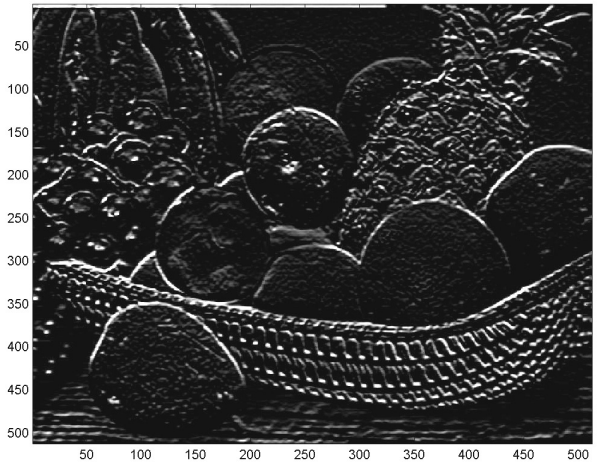
5 – La première partie du script ***filtrage_passehaut.m*** permet d'afficher le module de la fonction de transfert du filtre passe-haut que l'on souhaite utiliser. Ici, on considère le filtre de Prewitt vertical dont le noyau est :

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

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 :

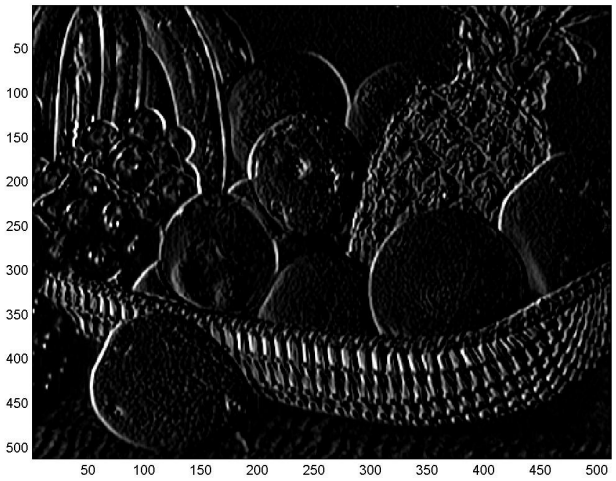


On a obtenu la détection des contours horizontaux de l'image.

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 verticaux :



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

Chapitre 3 – Filtrage Linéaire

TEST

On s'intéresse dans ce test à des filtres linéaires ***H*** décrits par leur noyau de convolution 2-D noté ***h***. La taille du noyau sera toujours 3 x 3, l'élément h(0,0) étant au centre du support du noyau. On appelle I_0 l'image d'entrée et I_S l'image de sortie. I_0 est une image monochrome à valeurs sur l'intervalle [0, 255].

1 – Quel est le noyau de convolution h_1 de taille 3 x 3 du **filtre identité** (tel que $I_S = I_0$) ?

2 – Soit le filtre H_2 dont le noyau de convolution h_2 est le suivant :

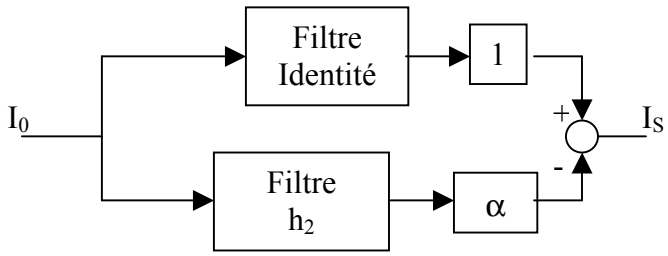
$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

2.1 – Quel est le gain en continu de ce filtre (gain à la fréquence spatiale $\nu_X = 0$ et $\nu_Y = 0$) ?

2.2 – Si, sur une zone d'image de taille au moins égale à 3 x 3, le signal est constant, égal à A, quelle est la valeur de I_S au centre de cette zone ?

3 – Déterminez, en écrivant le programme Matlab correspondant, le filtrage d'une image I_0 de taille (M x N) par le filtre linéaire de noyau h_2. L'image de sortie devra être de même taille que celle d'entrée I_0. Qu'observez-vous sur les bords des objets (tangon de pêche, filin, …) de l'image « ***Bateau*** » ?

4 – On veut améliorer l'image I_0 en accentuant les contrastes sur les bords des objets de l'image. Pour cela, on veut utiliser un troisième filtre équivalent fonctionnellement à faire la différence du filtre identité et d'une fraction (valeur α) du filtre de noyau h_2.



Écrire le programme Matlab construisant fonctionnellement ce troisième filtre. Observez les résultats obtenus pour les valeurs suivantes de α : 0 ; 1/10 ; 1/4 ; 1/2. Commentez les résultats obtenus sur la zone des filins et du tangon de pêche de l'image *Bateau*.