Prétraitement d'image

Convolution 1D (Signal)

Définition mathématique

Mathématiquement, la convolution est une opération mathématique qui prend deux signaux en entrée *h(t)* et *g(t)* et qui renvoie un nouveau signal $s(t)$, tel que :

$$s(t) = h(t) * g(t) = \int_{-\infty}^{+\infty} h(\tau) * g(t-\tau) d\tau$$

Comme la convolution opère sur deux variables d'entrée et qu'elle est très reliée à la multiplication, on utilise souvent le symbole $*$ pour symboliser cette opération :

$$s = h * g$$

Pour calculer un produit de convolution, il faut conserver le premier signal. Trouver le symétrique du second par rapport à l'axe des ordonnées puis décaler ce signal du temps t, multiplier le deux signaux obtenus et finalement intégrer le résultat.

Définition pour des signaux numériques

Signaux de même dimension

Si les signaux h et g sont numériques et définis par les suites {h(n)} et {g(n)} de dimension N, le produit de convolution s'écrit alors :

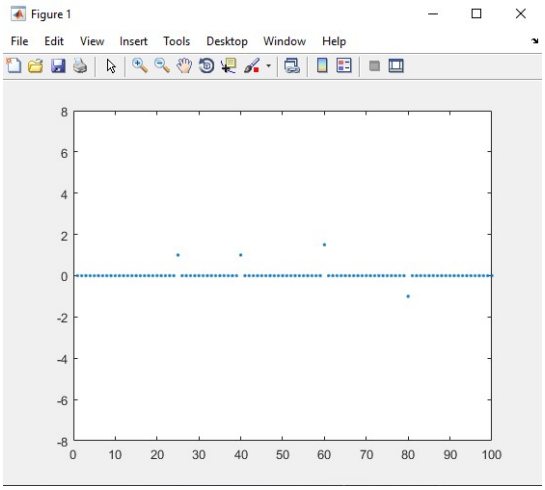
$$s(n) = h(n) * g(n) = \sum_{k=0}^{N-1} h(k) * g(n-k)$$

Les signaux *h* et *g* sont tous deux définis sur l'intervalle [*0, N-1*]. Toutefois dans l'opération de retournement du signal *g* l'indice *n-k* sera susceptible de courir de [-N+1*, N-1*]. Comme *g* n'est défini que sur l'intervalle [*0, N-1*] il importe de se fixer les règles de calcul et ces règles peuvent être différentes selon l'objectif que l'on se fixe.

Exemple :

1) Créer un signal quasiment nul partout, mais avec quelques valeurs non nulles par-ci par là :

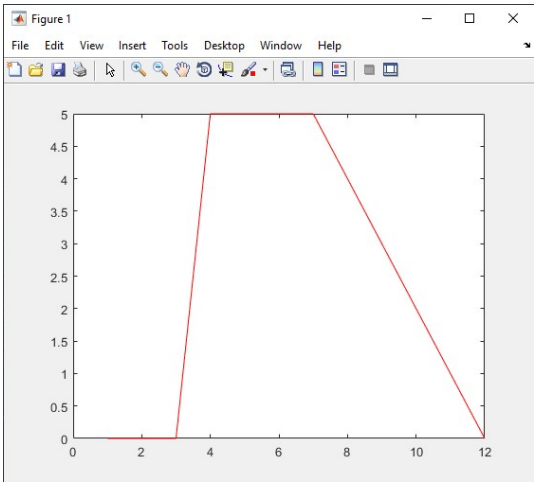
```
h = zeros(100,1); % Un vecteur avec que des zeros
h([25;40]) = 1; % on fixe la valeur de h à 1 pour les indices 55,128,302 et
50
h(60)=1.5; %h vaut 1.5 pour l'indice 60
h(80) = -1; %h vaut -1 en 80

plot(h,'.') % on regarde parce que c'est joli
axis([0 100 -8 8]) % fixe la zone de traçage (un peu comme xlim, et ylim en
une seule commande
```



Le signal h est non nul en quelques valeurs éparses.

2) Créer le deuxième signal, g , et je vous propose de lui donner une forme de trapèze :

```
g = [0;0;0;5;5;5;5;4;3;2;1;0];%g est un signal avec trois zéros, puis une
rampe qui descend de 5 jusqu'à zero
plot(g,'r')
```

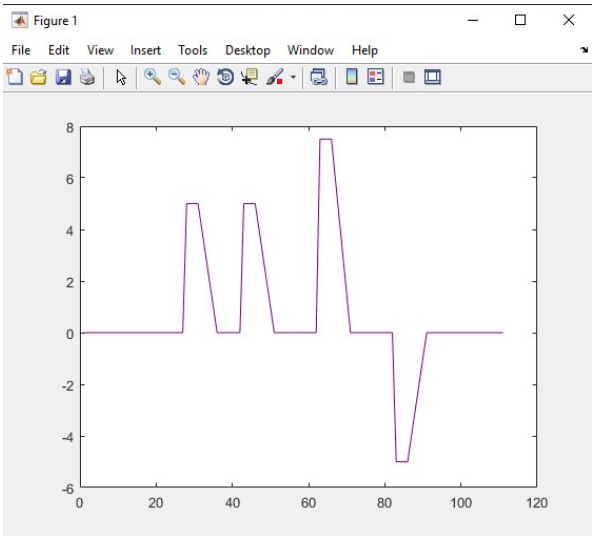


Le signal g a une forme de trapèze.

3) Faite la convolution de h avec g avec la fonction conv de MATLAB.

```
s = conv(h,g); %s est la convolution de h avec g
```

```
plot(s,'color',[0.5 0 0.5]); %On précise la couleur en RGB: 50% rouge, 50% bleu
```



La convolution en mauve.

Comment interpréter ce résultat ?

Le signal rouge a été recopié partout où le signal bleu était non nul. L'amplitude du recopiage dépend de la valeur du signal bleu. En particulier, si le signal bleu était négatif, le signal rouge est recopié vers le bas (négativement).

Convolution 2D (image)

Filtrage linéaire d'une image (signal 2D) :

Soit *I(x,y)* une matrice Bitmap, filtré une image signifie convoler une image *I(x,y)* avec une fonction appelé filtre f(x,y), cette convolution s'appelle réponse impulsionnelle du filtre.

a) Cas continu :

L'image filtrée est donnée par :

$$cov(x,y) = (f * I)(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x',y')I(x-x',y-y')dx'dy'$$

$$= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x-x',y-y')I(x',y')dx'dy'\text{K}$$

b) Cas discret :

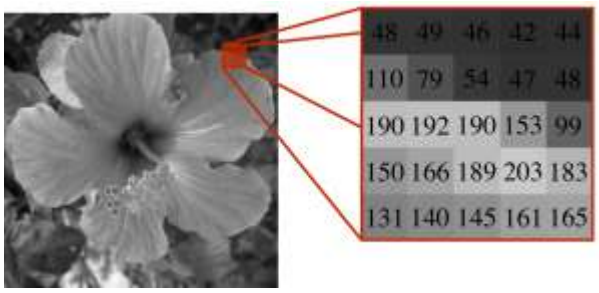
Soit [*-N/2, +N/2*] domaine borné de I et [*-K/2, +K/2*] domaine borné de *f*. On a nécessairement$K \leq N$, N étant la taille de l'image. Dans le cas discret la convolution s'écrit :

$$cov(x,y) = (f * I)(x,y) = \sum_{x'=-K/2}^{x'=+K/2} \sum_{y'=-N/2}^{y'=+N/2} f(x-x', y-y')I(x',y')$$

Remarque : le filtrage linéaire consiste simplement à remplacer chaque niveau de gris par une combinaison linéaire des niveaux de gris des points voisins ; les coefficients de cette combinaison linéaire sont définis par la réponse impulsionnelle du filtre. Cette réponse impulsionnelle est la réponse du filtre à la fonction impulsion

Exemple :

Soit l'image à niveau de gris suivante



On va extraire un carré rouge de taille 5 x 5 de l'image à gauche, et soit *I(x,y)* la matrice bitmap à droite correspondante qui contient la valeur numérique de chaque pixel et on veut convolée *I(x,y)* par le filtre linéaire $f'x', y') = \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$

Solution

Il faut crée une matrice de 7 x 7 en ajoutant deux lignes nuls, l'un au dessus et l'autre au dessous et deux colonnes nuls l'un à gauche et l'autre à droite. La matrice devient :

I' =

0	0	0	0	0	0	0
0	148	49	46	42	44	0
0	110	79	54	47	48	0
0	190	192	190	153	99	0
0	150	166	189	203	183	0
0	131	140	145	161	165	0
0	0	0	0	0	0	0

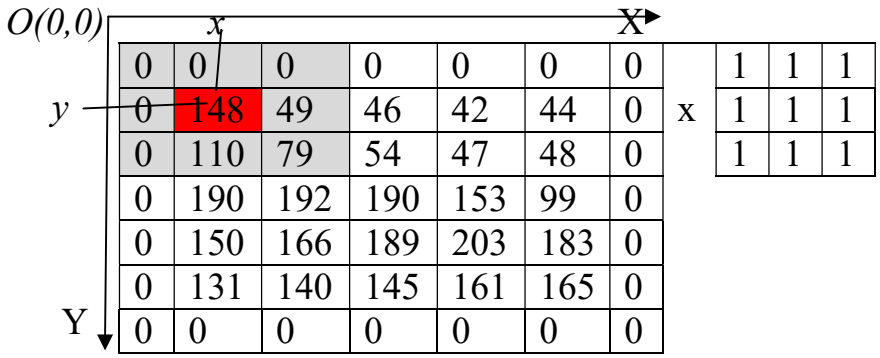
Maintenant, on va calculer le filtrage linéaire en multipliant la nouvelle matrice par le filtre

$f =$

1	1	1
1	1	1
1	1	1

On a ce schéma

Cov(x,y)=



0	0	0	0	0	0	0
0	148	49	46	42	44	0
0	110	79	54	47	48	0
0	190	192	190	153	99	0
0	150	166	189	203	183	0
0	131	140	145	161	165	0
0	0	0	0	0	0	0

x

1	1	1
1	1	1
1	1	1

On commence par la ligne (y)

Donc on va remplacer le pixel (x,y)=148 par le calcul de la moyenne de ces voisins 8-connexe ainsi que le pixel lui-même.

1)La nouvelle valeur du pixel rouge dont les coordonnées sont x, et y est :

I'(x,y)=(0 * 1+ 0* 1+ 0*1 +0*1 + 148*1 + 49*1+ 0*1+ 110*1 + 79*1)/9=386/9=42.8888…≈43.

2)On va appliqué le même filtre au pixel de coordonnée (x+1,y)=49

I'(x+1,y)=(0 * 1+ 0* 1+ 0*1 + 148*1 + 49*1+ 46*1+ 110*1 + 79*1+54*1)/9= 54.

3)On passe au point de coordonnée (x+2,y)=46

I'(x+2,y)=(0 * 1+ 0* 1+ 0*1 + 49*1+ 46*1+ 42*1+79*1+54*1+47*1)/9=317/9=35.2222….=35

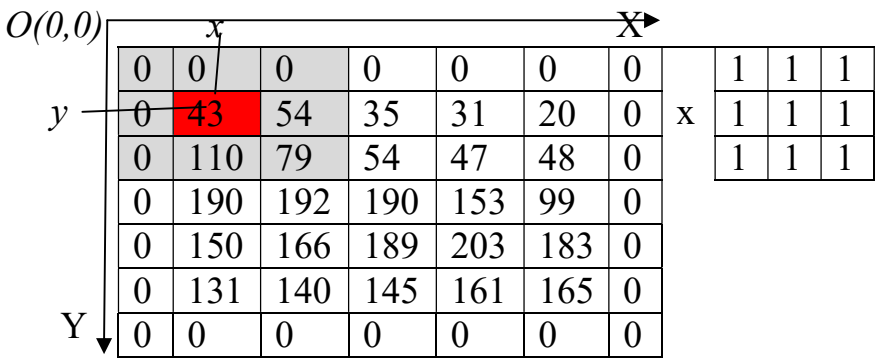
4)Au point suivant d coordonnée (x+3,y)=42

I'(x+3,y)=(0 * 1+ 0* 1+ 0*1 + 46*1+ 42*1+44*1+54*1+47*1+48*1)/9=281/9=31.222…..=31

5)Point de coordonnée (x+4,y)=44

I’(x+3,y)=(0 * 1+ 0* 1+ 0*1 +42*1+44*1+0*1+47*1+48*1+0*1)/9=281/9=181/9=20.1111…=20

Cov(x,y)=



0	0	0	0	0	0	0
0	43	54	35	31	20	0
0	110	79	54	47	48	0
0	190	192	190	153	99	0
0	150	166	189	203	183	0
0	131	140	145	161	165	0
0	0	0	0	0	0	0

x

1	1	1
1	1	1
1	1	1

On passe à ligne (y+1)

a)Point de coordonnée (x,y+1)=110

I’(x,y+1)=(0*1 + 148*1 + 49*1+ 0*1+ 110*1 + 79*1+0*1+190*1+192*1)/9=768/9=85.333…=85

b)Point de coordonnée (x+1,y+1)=79

I’(x+1,y+1)=(148*1 + 49*1+ 46*1+ 110*1 + 79*1+54*1+190*1+192*1+190*1)/9=1058/9=117.5555….=118

d)Point de coordonnée (x+2,y+1)=54

I’(x+2,y+1)=(49*1+ 46*1+ 42*1+79*1+54*1+47*1+192*1+190*1+153*1)/9= 852/9 = 94.666…=95

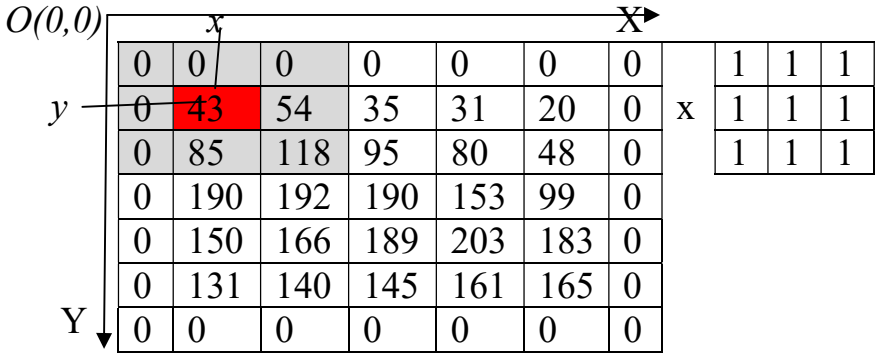
e)Point de coordonnée (x+3,y+1)=47

I’(x+3,y+1)=(46*1+ 42*1+44*1+54*1+47*1+48*1+190*1+153*1+99*1)/9= 723/9=80.333…=80

f)Point de coordonnée (x+4,y+1)=48

I’(x+4,y+1)=(42*1+44*1+0*1+47*1+48*1+0*1+153*1+99*1+0*1)/9=433/9=4 8.1111..=48

Cov(x,y)=



0	0	0	0	0	0	0		1	1	1
0	43	54	35	31	20	0	x	1	1	1
0	85	118	95	80	48	0		1	1	1
0	190	192	190	153	99	0				
0	150	166	189	203	183	0				
0	131	140	145	161	165	0				
0	0	0	0	0	0	0				

On passe à ligne (y+2)

I'(x,y+2)=(0*1 + 110*1 + 79*1+0*1+190*1+192*1+0*1+1*150+1*166)/9= 887/9=98.555555=99

I'(x+1,y+2)=(110*1 + 79*1+54*1+190*1+192*1+190*1+1*150+1*166+1*189)/9= 1320/9= 146.6666=147

I'(x+2,y+2)=(79*1+54*1+47*1+192*1+190*1+153*1+1*166+1*189+1*203)/9 = 1273/9= 141.44=141

I'(x+3,y+2)=(54*1+47*1+48*1+190*1+153*1+99*1+1*189+1*203+1*183)/9= 1166/9= 129.5555=130

I'(x+4,y+2)=(47*1+48*1+0*1+153*1+99*1+0*1+1*203+1*83+1*0)/9= 633/9=70.3333=70

On passe à ligne (y+3)

I'(x,y+3)=(0*1+190*1+192*1+0*1+1*150+1*166+1*0+1*131+1*140+1*145)/ 9= 922/9=102.4444=102

I'(x+1,y+3)=(190*1+192*1+190*1+1*150+1*166+1*189+1*131+1*140+1*14 5)/9= 1493/9=165.88888=166

I'(x+2,y+3)=(192*1+190*1+153*1+1*166+1*189+1*203+1*140+1*145+1*16 1)/9= 1539/9=171

I'(x+3,y+3)=(190*1+153*1+99*1+1*189+1*203+1*183+1*145+1*161+1*165)/9= 1488/9=165.33333=165

I’(x+4,y+3)=(153*1+99*1+0*1+1*203+1*83+1*0+1*161+1*165+1*0)/9= 864/9=96

On passe à ligne (y+4)

I’(x,y+4)=(0*1+1*150+1*166+1*0+1*131+1*140+1*0+1*0+1*0)/9= 587/9=102.3333=65.22222=65

I’(x+1,y+4)=(1*150+1*166+1*189+1*131+1*140+1*145+1*0+1*0+1*0)/9= 921/9=102

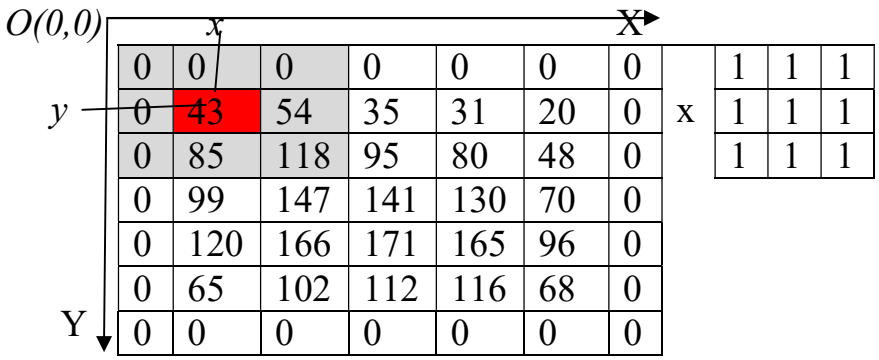
I’(x+2,y+4)=(1*166+1*189+1*203+1*140+1*145+1*161+1*0+1*0+1*0)/9= 1004/9= 111.5555=112

I’(x+3,y+4)=(1*189+1*203+1*183+1*145+1*161+1*165+1*0+1*0+1*0)/9= 1046/9=116.22222=116

I’(x+4,y+4)=(1*203+1*83+1*0+1*161+1*165+1*0+1*0+1*0+1*0)/9= 612/9=68

Le résultat du filtrage

Cov(x,y)=



0	0	0	0	0	0	0
0	43	54	35	31	20	0
0	85	118	95	80	48	0
0	99	147	141	130	70	0
0	120	166	171	165	96	0
0	65	102	112	116	68	0
0	0	0	0	0	0	0

x

1	1	1
1	1	1
1	1	1

//Pour Sobel Prewitt Roberts et Gaussien

Il existe d’autre masque de filtre pour filtrer une image, calcul de gradient selon X et le gradient selon Y

SobelX=

-1	0	1
-2	0	2
-1	0	1

SobelY=

-1	-2	1
0	0	0
1	2	1

RobertX=

0	0	0
0	1	0
0	0	-1

RobertY=

0	0	0
0	0	1
0	-1	0

PrewittX=

1	0	-1
1	0	1
1	0	-1

PrewittY=

1	1	1
0	0	0
-1	-1	1

//Convolution Sobel ou Roberts ou Prewitt

Algorithme

```
fonction  real Covolution(real  mat, real noyau[3][3], entier haut, entier larg)
{
```

// Déclaration des variables

```
 entier  i,j,x,y;  real temp;
 entier h=haut+2, l=larg+2;
 real bufferTemp [l] [h];

 real bufferT[l] [h];

 real buffer [l] [h];
```

//Début du programme

```
 Pour i=0 jusqu'a h  faire //h : hauteur de l'image
   bufferTemp[i][0]=0.0;bufferTemp[i][l-1]=0.0;

 Pour j=0 jusqu'à  l   faire // l : largeur de l'image
   bufferTemp[0][j]=0.0;bufferTemp[h-1][j]=0.0;

 Pour i=1 jusqu'à h-1 faire
      Pour j=1 jusqu'à  l-1 faire
          bufferTemp[i][j]=mat[i-1][j-1];

 Pour x=1 jusqu'a h-1 faire
     Pour y=1 jusqu'à  l-1 faire
    Début
        temp=0.0;
        Pour i=0 jusqu'a 3  faire
```

```
            Pour j=0 jusqu'à 3 faire
                  temp+=bufferTemp[x+i-1][y+j-1]*noyau[i][j];
          bufferT[x][y]=temp;
    Fin

Pour i=1 jusqu'a h-1 faire
   Pour j=1 jusqu'a l-1 faire
      buffer[i-1][j-1]=bufferT[i][j];


 retourner  buffer;
}
```

//Convolution Gaussienne

```
fonction  real  Covolution_Gaussien(real  mat, real  noyau[5][5] ], entier haut, entier larg)
{
```

//Déclaration des variables

```
 entier  i,j,x,y;  real  temp;
 entier  h=haut+4, l=larg+4;
 real bufferTemp [l] [h];
 real bufferT[l] [h];
 real buffer [larg] [haut];

 Pour i=0 jusqu'a l faire
 {
   bufferTemp[i][0]=0.0;bufferTemp[i][1]=0.0;bufferTemp[i][h-2]=0.0;bufferTemp[i][h-1]=0.0;
 }
 Pour j=0 jusqu'a h  faire
 {
   bufferTemp[0][j]=0.0;bufferTemp[1][j]=0.0;bufferTemp[l-2][j]=0.0;bufferTemp[h-1][j]=0.0;
 }

 Pour i=2 jusqu'a h-2  faire
    Pour j=2 jusqu'a  l-2  faire
         bufferTemp[i][j]=mat[i-2][j-2];

 Pour x=1  jusqu'a h-4 faire
   Pour y=1 jusqu'à l-4  faire
  Début
     temp=0.0;
     Pour i=0 jusqu'a 5  faire
        Pour j=0 jusqu'a 5 faire
           temp+=bufferTemp[x+i-1][y+j-1]*((1.0/115.0)*noyau[i][j]);
           bufferT[x][y]=temp;
 fin
```

```
Pour i=2 jusqu'a h-2 faire
    Pour j=2 jusqu'à l-2  faire
       buffer[i-2][j-2]=bufferT[i][j];

 retourner  buffer;
}
```

Programme en c

```
const float Sobelx[3][3]={{-1.0,0.0,1.0},{-2.0,0.0,2.0},{-1.0,0.0,1.0}} ;
const float Sobely[3][3]={{-1.0,-2.0,-1.0},{0.0,0.0,0.0},{1.0,2.0,1.0}} ;

const float Robertsx[3][3]={{0.0,0.0,0.0},{0.0,1.0,0.0},{0.0,0.0,-1.0}};
const float Robertsy[3][3]={{0.0,0.0,0.0},{0.0,0.0,1.0},{0.0,-1.0,0.0}};

const float Prewittx[3][3]={{1.0,0.0,-1.0},{1.0,0.0,-1.0},{1.0,0.0,-1.0}};
const float Prewitty[3][3]={{1.0,1.0,1.0},{0.0,0.0,0.0},{-1.0,-1.0,-1.0}};

const  float
Gaussien[5][5]={{2.0,4.0,5.0,4.0,2.0},{4.0,9.0,12.0,9.0,4.0},{5.0,12.0,15.0,12.0,5.0},{4.0,9.0,12.0,9.0
,4.0},{2.0,4.0,5.0,4.0,2.0}};

float ** Covolution(float ** mat, const float noyau[3][3])
{
 int i,j,x,y;
 int h=haut+2, l=larg+2;
 float ** bufferTemp;
 bufferTemp=new float *[h];
 *bufferTemp=new float[h*l];
 for(i=0;i<h;++i) bufferTemp[i]=*bufferTemp+i*l;

 float ** bufferT;
 bufferT=new float *[h];
 *bufferT=new float[h*l];
 for(i=0;i<h;++i) bufferT[i]=*bufferT+i*l;

 float ** buffer;
 float temp;
 buffer=new float *[haut];
 *buffer=new float[haut*larg];
 for(i=0;i<haut;++i) buffer[i]=*buffer+i*larg;

 for(i=0;i<h;++i)
   bufferTemp[i][0]=0.0;bufferTemp[i][l-1]=0.0;

 for(j=0;j<l;++j)
   bufferTemp[0][j]=0.0;bufferTemp[h-1][j]=0.0;
```

```
 for(i=1;i<h-1;++i)
          for(j=1;j<l-1;++j)
             bufferTemp[i][j]=mat[i-1][j-1];

 for(x=1;x<h-1;++x)
   for(y=1;y<l-1;++y)
{
     temp=0.0;
     for(i=0;i<3;++i)
         for(j=0;j<3;++j)
           temp+=bufferTemp[x+i-1][y+j-1]*noyau[i][j];
          bufferT[x][y]=temp;
}
for(i=1;i<h-1;++i)
  for(j=1;j<l-1;++j)
     buffer[i-1][j-1]=bufferT[i][j];
 return buffer;
}
```

//Convolution Gaussienne

```
float ** Covolution_Gaussien(float ** mat, const float noyau[5][5])
{
 int i,j,x,y;
 int h=haut+4, l=larg+4;
 float ** bufferTemp;
 bufferTemp=new float *[h];
 *bufferTemp=new float[h*l];
 for(i=0;i<h;++i) bufferTemp[i]=*bufferTemp+i*l;

 float ** bufferT;
 bufferT=new float *[h];
 *bufferT=new float[h*l];
 for(i=0;i<h;++i) bufferT[i]=*bufferT+i*l;

 float ** buffer;
 float temp;
 buffer=new float *[haut];
 *buffer=new float[haut*larg];
 for(i=0;i<haut;++i)    buffer[i]=*buffer+i*larg;
 for(i=0;i<h;++i)
 {
   bufferTemp[i][0]=0.0;bufferTemp[i][1]=0.0;bufferTemp[i][h-2]=0.0;bufferTemp[i][h-1]=0.0;
 }
 for(j=0;j<h;++j)
 {
   bufferTemp[0][j]=0.0;bufferTemp[1][j]=0.0;bufferTemp[l-2][j]=0.0;bufferTemp[h-1][j]=0.0;
 }
```

```
 for(i=2;i<h-2;++i)
    for(j=2;j<l-2;++j)
         bufferTemp[i][j]=mat[i-2][j-2];
 for(x=1;x<h-4;++x)
  for(y=1;y<l-4;++y)
  {
    temp=0.0;
    for(i=0;i<5;++i)
       for(j=0;j<5;++j)
           temp+=bufferTemp[x+i-1][y+j-1]*((1.0/115.0)*noyau[i][j]);
           bufferT[x][y]=temp;
}
for(i=2;i<h-2;++i)
   for(j=2;j<l-2;++j)
      buffer[i-2][j-2]=bufferT[i][j];
 return buffer;
}
```

Exercice :

Soit l'image « im » suivante :

im=

145	145	145	0	0	0	120	2
145	145	145	1	1	120	120	120
145	145	145	2	4	7	120	3
145	145	145	7	3	8	0	10
110	112	123	4	9	9	20	11

Question : appliquer sur cette image le filtrage de Sobel , Roberts et Gaussian.

Travaux pratiques

1) Ecrire une fonction MATLAB qui permet de faire la convolution d'une image par un masque Noyau de dimension 3 x3.
2) Ecrire une autre fonction MATLAB qui permet de faire le filtrage d'une image par un masque gaussienne 5x 5.