Lab 3 IPT 2021-2022

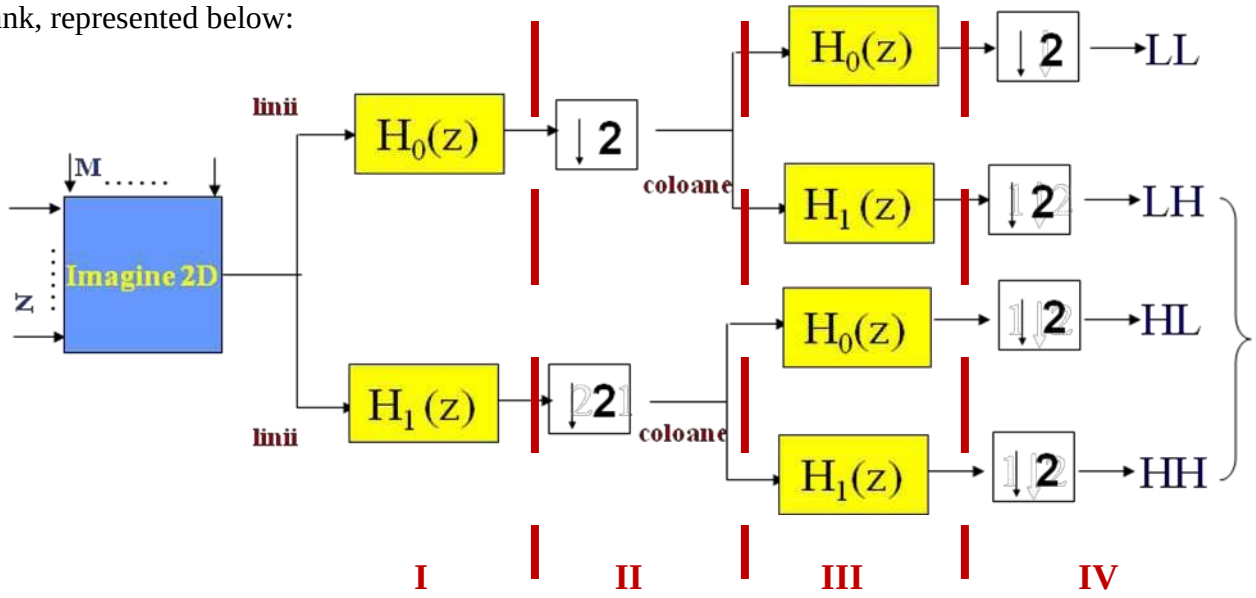
The discrete wavelet transform (DWT). Blackfin BF533- based implementation 1. Introduction

The continuous wavelet transform (CWT) allows analysis, interpretation and processing of nonstationary signals and images using multi-resolution techniques that are able to represent the input signal at multiple scales. The numerical approximation for the CWT is called DWT (discrete wavelet transform) and its implementation is done in practice using analysis and synthesis filter banks [1]. The filter coefficients must be chosen in order to fulfill the perfect reconstruction condition.

The purpose of the lab is to study an implementation of the forward and inverse DWT using filter banks. The programming language used is C and the development environment is Visual DSP ++.

2. The forward DWT transform

As mentioned previously the forward DWT transform can be implemented using an analysis filter bank, represented below:



The two orthogonal analysis filters H0 and H1 are respectively low pass and high pass filters. The steps involved in the direct DWT computation are listed below:

- filtering the lines with filters H0 and H1 (I)
- decimation of the two resulting images by odd columns elimination (II)
- filtering the columns with filters H0 and H1 of all the images resulting from the previous step (III)
- decimation of the four resulting subimages by removing the odd lines (IV)

Appendix 1 includes a sequence of code that can be used for implementing the direct DWT transform. The computations involved in the computation of the LH (LowHigh) image are explained below:

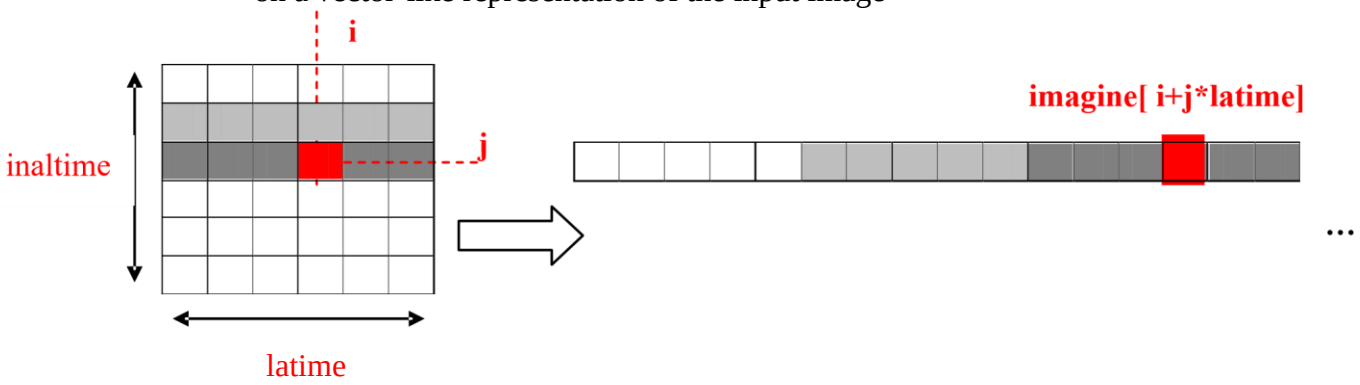
- variable definition for intermediate results ***float imagAnaliza[16384];*** - the result of the analysis filter bank
  float lis[16384]; - first filtering step
  float dec_2[16384]; - first decimation step

float dec_2_2[16384]; - second filtering step
float fil_dec_2[16384]; - second filtering step

- implementation is done on IV stages. The effective implementation of the operations will be explained only for the LH subband.
- the C code associated to the filtering operation taking place on the lines of the input image is listed below:

```
//////////////////////LH//////////////////////////////////////////////////
    //Filtrare pe linii filtru H0 for (i=0; i<latime-1;i++) for (j=0; j<inaltime;j++)
    lis[i+j*latime]=(ent1[i+j*latime]+ent1[(i+1)+j*latime])*0.707106781;
```

-the employed low pass filter is of Haar type H_0 [*0.707106781; 0.707106781*] and it operates on a vector-like representation of the input image



- the C code associated to the decimation operation taking place on the columns of the filtered image is the following (only even numbered columns are retained):

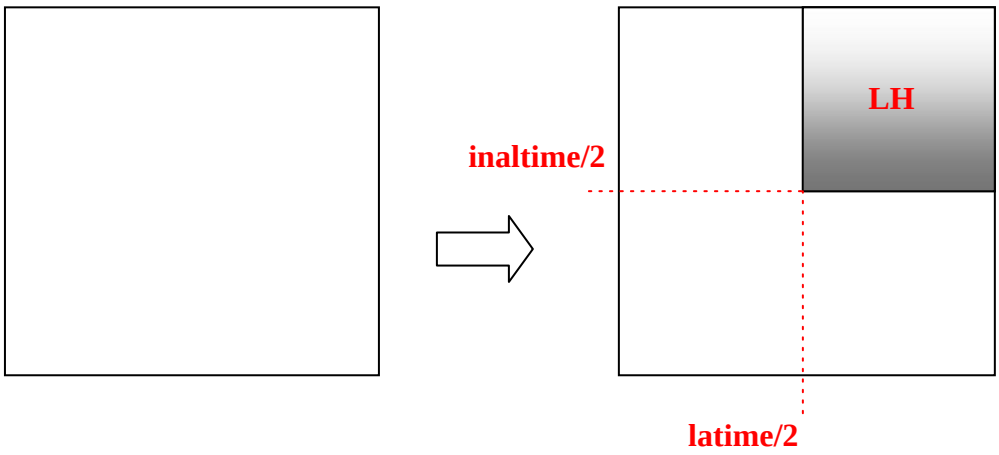
```
for (i=0;i<latime/2;i++) for (j=0; j<inaltime;j++)
        dec_2[i+j*latime]=lis[2*i+j*latime];
```

-the same procedure is repeated in stages III and IV with the filter H_1[*0.707106781; - 0.707106781*] applied on the columns and decimation taking place on lines.

```
//Filtrare coloane cu filtru H1 for (i=0; i<latime/2;i++) for (j=0; j<inaltime-1;j++)
    lis_dec_2[i+j*latime]=(dec_2[i+j*latime]-dec_2[i+(j+1)*latime])*0.707106781;

    //Decimare linii for (i=0;i<latime/2;i++) for (j=0;
    j<inaltime/2;j++)
    dec_2_2[i+j*latime]=lis_dec_2[i+2*j*latime];
```

- the resulting image is placed on the output image *ieșire* at predefined coordinates, as shown below:



- the C code that places the LH image starting from the decimation result is the following:

```
for (i=latime/2; i<latime;i++)
for (j=0; j<inaltime;j++)
        iesire[i+j*latime]=dec_2_2[i-latime/2+j*latime];
```

3. Exercises

1) The **Wavelet.dpj** Visual DSP project located in the archive placed on the website http://ares.utcluj.ro/tpi implements the code listed in Appendix 1.

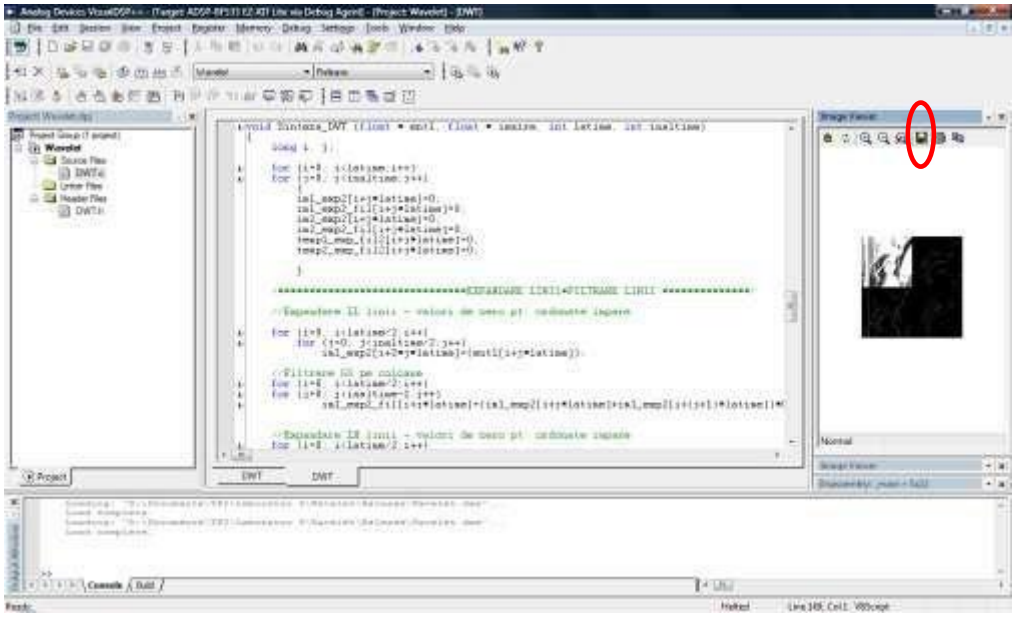
2) Identify the method of calculation and mapping of the other images (LL, HL, HH) in the output image.

3) Use the Image Viewer tool for initializing the input variable with the **lena.bmp** image and view the result of the analysis stage.

4) Save the resulting image for future reference using the *Image Viewer* tool .

5) Repeat the previous steps for the **test.bmp** image from the same directory.

6) How can the DWT coefficients can be interpreted?

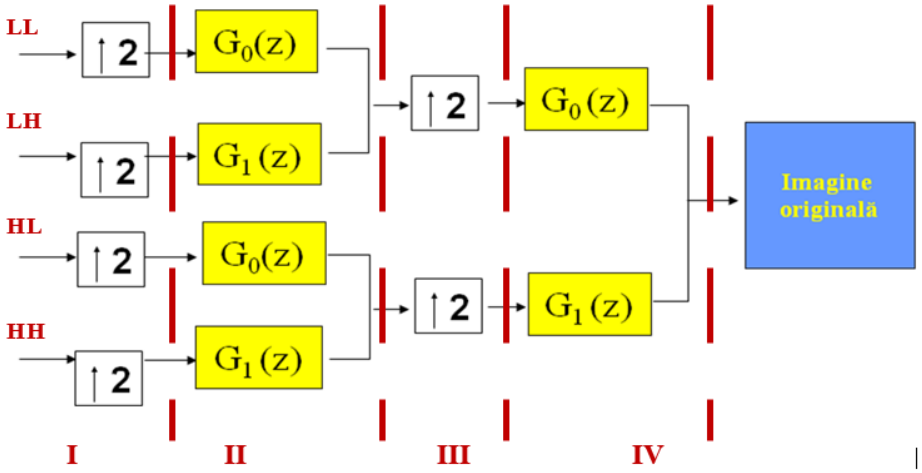


7) Write a C function that will set the coefficients of the LH, HL and HH images to zero.

8) Which are the modification that must be brought to the application for allowing a forward DWT computation on 2 decomposition levels?

4. Inverse DWT computation

The inverse DWT transform is computed using a synthesis filter bank :



The steps associated inverse DWT transform are being defined as being complementary to those of the analysis stage:

- expansion with two taking place on the lines (I)
- column filtering of the results using filters G0 and G1 (II) followed by summation
- expansion with two taking place on the two resulting images (III), filtering with G0 and G1 followed by summation

5. Exercises

The project **Wavelet.dpj** includes also the code associated to the inverse DWT transform using a synthesis filter bank. To view the combined results of the analysis / synthesis uncomment the line

/ / Sinteza_DWT (imagAnaliza, imagSinteza, 128,128);

placed under the body of the *main* function of the project and comment the code associated with the visualization part of the analysis stage.

1)Based on the code included in Appendix 2 identify how the inverse DWT transform is implemented

2)Run the application and view the results of the analysis and synthesis stages. Explain the differences between image the input and the output images.

3)What synthesis filter should be employed for allowing perfect reconstruction to take place (modulo quantization errors of the coefficients)? Change accordingly the filter coefficients. They are declared as [g0_0; g0_1] for the G0 filter and, respectively, by [g1_0; g1_0] for the G1 filter.

4)Check if, with the modified coefficients, the perfect reconstruction condition is fulfilled 5) Integrate the changes made as indicated in § 3 step 4) with the previous ones and explain the result.

References

[1] R. Terebes – Tehnologii de prelucrare a informației – lecture notes 2022-2023, http://ares.utcluj.ro/tpi

Appendix 1.Forward DWT computation

```
void Analiza_DWT (unsigned char * ent1, float * iesire, int latime, int inaltime)
{
int i, j;

//////////////////////LL//////////////////////////////////////////////////
//Filtrare  pe linii filtru H0
for (i=1; i<latime;i++)
   for (j=0; j<inaltime;j++)
          fil[i+j*latime]=(ent1[i+j*latime]+ent1[(i-1)+j*latime])*0.707106781;

//Decimare coloane
for (i=0;i<latime/2;i++)
   for (j=0; j<inaltime;j++)
          dec_2[i+j*latime]=fil[2*i+j*latime];

//Filtrare coloane cu filtru H0
for (i=0; i<latime/2;i++)
   for (j=1; j<inaltime;j++)
          fil_dec_2[i+j*latime]=(dec_2[i+j*latime]+dec_2[i+(j-1)*latime])*0.707106781;

//Decimare linii
for (i=0;i<latime/2;i++)
   for (j=0; j<inaltime/2;j++)
          dec_2_2[i+j*latime]=fil_dec_2[i+2*j*latime];

// Construire imagine iesire
for (i=0; i<latime;i++)
   for (j=0; j<inaltime;j++)
          iesire[i+j*latime]=dec_2_2[i+j*latime];

// Golire imagini temporare
   for (i=0; i<latime;i++)
   for (j=0; j<inaltime;j++)
   {
          fil[i+j*latime]=0;
          fil_dec_2[i+j*latime]=0;
          dec_2[i+j*latime]=0;
          dec_2_2[i+j*latime]=0;
   }
//////////////////////LH//////////////////////////////////////////////////
//Filtrare  pe linii filtru H0
for (i=1; i<latime;i++)
   for (j=0; j<inaltime;j++)
          fil[i+j*latime]=(ent1[i+j*latime]+ent1[(i-1)+j*latime])*0.707106781;

//Decimare coloane
for (i=0;i<latime/2;i++)
   for (j=0; j<inaltime;j++)
          dec_2[i+j*latime]=fil[2*i+j*latime];

//Filtrare coloane cu filtru H1
for (i=0; i<latime/2;i++)
   for (j=1; j<inaltime;j++)
          fil_dec_2[i+j*latime]=(dec_2[i+j*latime]-dec_2[i+(j-1)*latime])*0.707106781;

//Decimare linii
for (i=0;i<latime/2;i++)
   for (j=0; j<inaltime/2;j++)
          dec_2_2[i+j*latime]=fil_dec_2[i+2*j*latime];
```

```
// Construire imagine iesire
for (i=latime/2; i<latime;i++)
   for (j=0; j<inaltime;j++)
          iesire[i+j*latime]=dec_2_2[i-latime/2+j*latime];

// Golire imagini temporare
   for (i=0; i<latime;i++)
   for (j=0; j<inaltime;j++)
   {
          fil[i+j*latime]=0;
          fil_dec_2[i+j*latime]=0;
          dec_2[i+j*latime]=0;
          dec_2_2[i+j*latime]=0;
   }
//////////////////////HL////////////////////////////////////////////////
//Filtrare  pe linii filtru H1
for (i=1; i<latime;i++)
   for (j=0; j<inaltime;j++)
          fil[i+j*latime]=(ent1[i+j*latime]-ent1[(i-1)+j*latime])*0.707106781;

//Decimare coloane
for (i=0;i<latime/2;i++)
   for (j=0; j<inaltime;j++)
          dec_2[i+j*latime]=fil[2*i+j*latime];

//Filtrare coloane cu filtru H0
for (i=0; i<latime/2;i++)
   for (j=1; j<inaltime;j++)
          fil_dec_2[i+j*latime]=(dec_2[i+j*latime]+dec_2[i+(j-1)*latime])*0.707106781;

//Decimare linii
for (i=0;i<latime/2;i++)
   for (j=0; j<inaltime/2;j++)
          dec_2_2[i+j*latime]=fil_dec_2[i+2*j*latime];

// Construire imagine iesire
for (i=0; i<latime;i++)
   for (j=inaltime/2; j<inaltime;j++)
          iesire[i+j*latime]=dec_2_2[i+(j-inaltime/2)*latime];

// Golire imagini temporare
   for (i=0; i<latime;i++)
   for (j=0; j<inaltime;j++)
   {
          fil[i+j*latime]=0;
          fil_dec_2[i+j*latime]=0;
          dec_2[i+j*latime]=0;
          dec_2_2[i+j*latime]=0;
   }

//////////////////////HH////////////////////////////////////////////////
//Filtrare  pe linii filtru H1
for (i=1; i<latime;i++)
   for (j=0; j<inaltime;j++)
          fil[i+j*latime]=(ent1[i+j*latime]-ent1[(i-1)+j*latime])*0.707106781;

//Decimare coloane
for (i=0;i<latime/2;i++)
   for (j=0; j<inaltime;j++)
```

```
          dec_2[i+j*latime]=fil[2*i+j*latime];

//Filtrare coloane cu filtru H0
for (i=0; i<latime/2;i++)
   for (j=1; j<inaltime;j++)
          fil_dec_2[i+j*latime]=(dec_2[i+j*latime]-dec_2[i+(j-1)*latime])*0.707106781;

//Decimare linii
for (i=0;i<latime/2;i++)
   for (j=0; j<inaltime/2;j++)
          dec_2_2[i+j*latime]=fil_dec_2[i+2*j*latime];

// Construire imagine iesire
for (i=latime/2; i<latime;i++)
   for (j=inaltime/2; j<inaltime;j++)
          iesire[i+j*latime]=dec_2_2[i-latime/2+(j-inaltime/2)*latime];

//////////////////////////DOAR PT VIZUALIZARE//////////////////////////
for (i=0;i<latime;i++)
for (j=0;j<inaltime;j++)
{
   if (iesire[i+j*latime]>255)
          iesire[i+j*latime]=255;
   if (iesire[i+j*latime]<0)
          iesire[i+j*latime]=0;
   imagOut[i+j*latime]= (unsigned char)iesire[i+j*latime];
}////////////////////////// SFARSIT VIZUALIZARE//////////////////////////
}
```

Appendix 2. Inverse DWT computation

```
void Sinteza_DWT (float * ent1, float * iesire, int latime, int inaltime)
{
long i, j;
////////////////////// DEFINIRE COEFICIENTI//////////////////////////
float g0_0 =1;
float g0_1 =1;
float g1_0 =1;
float g1_1 =1;

////////////////////// SFARSIT DEFINIRE//////////////////////////////

for (i=0; i<latime;i++)
   for (j=0; j<inaltime;j++)
   {
   im1_exp2[i+j*latime]=0;
   im1_exp2_fil[i+j*latime]=0;
   im2_exp2[i+j*latime]=0;
   im2_exp2_fil[i+j*latime]=0;
   temp1_exp_fil2[i+j*latime]=0;
   temp2_exp_fil2[i+j*latime]=0;

   }

/*****************************EXPANDARE LINII*FILTRARE LINII **************/

//Expandare LL linii - valori de zero pt. ordonate impare

for (i=0; i<latime/2;i++)
for (j=0; j<inaltime/2;j++)
   im1_exp2[i+2*j*latime]=(ent1[i+j*latime]);

//Filtrare G0 pe coloane
for (i=0; i<latime/2;i++)
for (j=1; j<inaltime;j++)
   im1_exp2_fil[i+j*latime]=(g0_0*im1_exp2[i+j*latime]+g0_1*im1_exp2[i+(j-1)*latime]);


//Expandare LH linii - valori de zero pt. ordonate impare
for (i=0; i<latime/2;i++)
for (j=0; j<inaltime/2;j++)
   im2_exp2[i+2*j*latime]=ent1[i+latime/2+(j)*latime];

//Filtrare G1 pe coloane
for (i=0; i<latime/2;i++)
for (j=1; j<inaltime;j++)
   im2_exp2_fil[i+j*latime]=(g1_0*im2_exp2[i+j*latime]+g1_1*im2_exp2[i+(j-1)*latime]);


//Adunare imagini temporare
for (i=0; i<latime/2;i++)
for (j=0; j<inaltime;j++)
   temp1_exp_fil2[i+j*latime]=im1_exp2_fil[i+j*latime]+im2_exp2_fil[i+j*latime];

//Golire imagini temporare
for (i=0; i<latime;i++)
for (j=0; j<inaltime;j++)
   {
          im1_exp2[i+j*latime]=0;
          im1_exp2_fil[i+j*latime]=0;
```

```
          im2_exp2[i+j*latime]=0;
          im2_exp2_fil[i+j*latime]=0;
   }

//Expandare HL -valori de zero pt. ordonate impare
for (i=0; i<latime/2;i++)
   for (j=0; j<inaltime/2;j++)
          im1_exp2[i+2*j*latime]=(ent1[i+(j+latime/2)*latime]);

//Filtrare G0 pe coloane
for (i=0; i<latime/2;i++)
for (j=1; j<inaltime;j++)
   im1_exp2_fil[i+j*latime]=(g0_0*im1_exp2[i+j*latime]+g0_1*im1_exp2[i+(j-1)*latime]);


//Expandare HH coloane -valori de zero pt. ordonate impare
for (i=0; i<latime/2;i++)
   for (j=0; j<inaltime/2;j++)
          im2_exp2[i+2*j*latime]=(ent1[i+latime/2+(j+latime/2)*latime]);

//Filtrare G1 pe coloane
for (i=0; i<latime/2;i++)
for (j=1; j<inaltime;j++)
   im2_exp2_fil[i+j*latime]=(g1_0*im2_exp2[i+j*latime]+g1_1*im2_exp2[i+(j-1)*latime]);

//Adunare imagini temporare
for (i=0; i<latime/2;i++)
for (j=0; j<inaltime;j++)
          temp2_exp_fil2[i+j*latime]=im1_exp2_fil[i+j*latime]+im2_exp2_fil[i+j*latime];

// Golire imagini intermediare
   for (i=0; i<latime;i++)
   for (j=0; j<inaltime;j++)
   {
          im1_exp2[i+j*latime]=0;
          im2_exp2[i+j*latime]=0;
          im1_exp2_fil[i+j*latime]=0;
          im2_exp2_fil[i+j*latime]=0;
   }


//EXPANDARE pe coloane - valori de zero pentru abscise imparre


for (i=0; i<latime/2;i++)
   for (j=0; j<inaltime;j++)
          im1_exp2[2*i+j*latime]=(temp1_exp_fil2[i+j*latime]);

//Filtrare G0 pe linii
for (i=1; i<latime;i++)
for (j=0; j<inaltime;j++)
   im1_exp2_fil[i+j*latime]=(g0_0*im1_exp2[i+j*latime]+g0_1*im1_exp2[i-1+(j)*latime]);



//EXPANDARE pe coloane - valori de zero pentru abscise impare
for (i=0; i<latime/2;i++)
   for (j=0; j<inaltime;j++)
          im2_exp2[2*i+j*latime]=(temp2_exp_fil2[i+j*latime]);
```

```
//Filtrare G1 pe linii
for (i=1; i<latime;i++)
for (j=0; j<inaltime;j++)
   im2_exp2_fil[i+j*latime]=(g1_0*im2_exp2[i+(j)*latime]+g1_1*im2_exp2[i-1+j*latime]);

for (i=0; i<latime;i++)
   for (j=0; j<inaltime;j++)
                        iesire[i+j*latime]=im1_exp2_fil[i+j*latime]+im2_exp2_fil[i+j*latime];

   for (i=0;i<latime;i++)
for (j=0;j<inaltime;j++)
{
   if (iesire[i+j*latime]>255)
          iesire[i+j*latime]=255;
   if (iesire[i+j*latime]<0)
          iesire[i+j*latime]=0;
   imagOut[i+j*latime]= (unsigned char)iesire[i+j*latime];
}
}
```