**Homework -6**

**Image Compression using Biorthogonal Coiflets**

**Abstract**

The main idea of this homework is to implement Biorthogonal Coiflets filter bank for compression of image. The biorthogonal Coiflets filter banks are used for perfect reconstruction at the receiver side. The low-pass and high-pass filters in this project is used from the paper Biorthogonal Coiflets. The MATLAB code for the implementation is attached as a .m file.

**MATLAB implementation**

Here is the MATLAB code for the implementation of two stages of filter banks.

%% Image Compression using Biorthogonal coiflets

inp=imread('1829422454.jpg'); % Input image

inp\_image = rgb2gray(inp); %Converted to grayscale

figure;

subplot(1,2,1),imshow(inp); title("Input image");

subplot(1,2,2);imshow(inp\_image); title("Input gray scale image");

The above code inputs an image and converts the image to gray scale. Both the images are plotted in a figure. Figure 1 depicts the input image.

A picture containing text

Description automatically generated

Figure 1: Input images

%% Even length filter bank

low = [-0.000379700958235, 0.00215163876333, -0.00198612808923,...

-0.01173621143634, 0.0286885168444, 0.01475409437712, ...

-0.1290983258, 0.0946721055865, 0.7100407918988, ...

0.7100407918988, 0.0946721055865, -0.1290983258, ...

0.01475409437712, 0.0286885168444, -0.01173621143634, ...

-0.00198612808923, 0.00215163876333, -0.000379700958235];

high = [-3.98960956074\*10^(-6), 0.00002260778751086, 6.3914743024\*10^(-6), -0.000277789678273, ...

0.000345091651368, 0.00155825777815, -0.00360374050808, -0.00499203440572, ...

0.02440592511743, 0.02334136673386, -0.106073569652, -0.1467785003718, 0.66490458055676, ...

0.66490458055676, -0.1467785003718, -0.106073569652, 0.02334136673386, 0.02440592511743, ...

-0.00499203440572, -0.00360374050808, 0.00155825777815, 0.000345091651368, ...

-0.000277789678273, 6.3914743024\*10^(-6), 0.00002260778751086, -3.98960956074\*10^(-6)];

The filter coefficients are initialized to the ones that are present in the paper. For the experiment I have used even length filter.

%% Compression

%% First stage processing rows

[row,col]=size(inp\_image); %Calculating size of input image

trend1=[]; dev1=[];

for i=1:1:row

temp=inp\_image(i,:); %each row of image

res\_trend=cconv(temp,low,col); %Circular convolution with low pass filter

trend1=[trend1;downsample(res\_trend,2)]; %Downsampling by a factor of 2

res\_dev=cconv(temp,high,col); %Circular convolution with high pass filter

dev1=[dev1;downsample(res\_dev,2)]; %Downsampling by a factor of 2

end

trend1=cast(real(trend1),'uint8'); %Converting double to uint8

dev1=cast(real(dev1),'uint8'); %Converting double to uint8

figure;

subplot(1,2,1),imshow(trend1);title("Lowpass filtered");

subplot(1,2,2),imshow(dev1);title("Highpass filtered");

Graphical user interface

Description automatically generated with medium confidence

Figure 2: Stage 1 compressed images

In figure 2, the images after stage 1 compression is obtained. In stage 1, the rows in the image are processed. Each row is circularly convolved with either low pass or high pass filter and downsampled. The resulting two images after row processing is shown in figure 2.

%% Second stage processing columns

trend12=[];trend21=[];dev12=[];dev21=[];

[row,col]=size(trend1); %Dividing trend image from first stage to two images

for i =1:1:col

temp=trend1(:,i); %each column of the image

res\_trend=cconv(temp,low,row);%Circular convolution with low pass filter

trend12=[trend12,downsample(res\_trend,2)];%Downsampling by a factor of 2

res\_dev=cconv(temp,high,row);%Circular convolution with high pass filter

dev12=[dev12,downsample(res\_dev,2)];%Downsampling by a factor of 2

end

[row,col]=size(dev1); %Dividing deviation image from first stage to two images

for i =1:1:col

temp=dev1(:,i); %each column of the image

res\_trend=cconv(temp,low,row);%Circular convolution with low pass filter

trend21=[trend21,downsample(res\_trend,2)];%Downsampling by a factor of 2

res\_dev=cconv(temp,high,row);%Circular convolution with high pass filter

dev21=[dev21,downsample(res\_dev,2)];%Downsampling by a factor of 2

end

%%

trend12=cast(real(trend12),'uint8');%Converting double to uint8

dev12=cast(real(dev12),'uint8');%Converting double to uint8

trend21=cast(real(trend21),'uint8');%Converting double to uint8

dev21=cast(real(dev21),'uint8');%Converting double to uint8

%%

figure;

subplot(2,2,1),imshow(trend12);title("Lowpass-Lowpass filtered");

subplot(2,2,2),imshow(trend21);title("Lowpass-Highpass filtered");

subplot(2,2,3),imshow(dev12);title("Highpass-Lowpass filtered");

subplot(2,2,4),imshow(dev21);title("Highpass-Highpass filtered");

Stage 2 processes two stage 1 images with respect to their columns. Each image is divided into 2 and hence stage 2 has 4 images, first one being lowpass- lowpass filtered, second one being lowpass-highpass filtered. Third and fourth images are resultant of second image column processing. They are high-low filtered and high-pass high-pass filtered. Figure 3 represents the four resulting compressed images that can be used to reconstruct the input image perfectly. In the edges of these images there exists an edge problem that is due to the circular convolution of filter and image. There are other techniques to mitigate this.

A picture containing old

Description automatically generated

Figure 3: Stage 2 final output