**NATIONAL INSTITUTE OF TECHNOLOGY CALICUT**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

****

**EC 4091: DIGITAL SIGNAL PROCESSING LABORATORY**

**LAB PROJECT**

**Filter Banks**

Date: 10-10-2018

**Submitted by:**

B.Veda Sree - B150734EC

B.Sravani - B150783EC

Ch.Priyanka - B150837EC

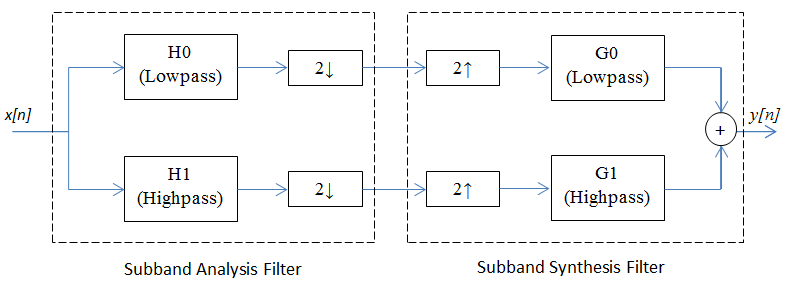
Ch.Vamsi Narayana Reddy - B150530EC

**Aim:** To design and implement a filter bank with two channels for the reconstruction of audio, black and white and color images.

**Theory:**

In Digital Signal Processing often there is a need to decompose signals into low and high frequency bands, after which they need to be combined to reconstruct the original signal. Such an example is found in sub-band coding (SBC).The 2-channel filter bank is also known as Quadrature Mirror Filter (QMF) Banks since they use power complementary filters.

The block diagram for 2-channel filter bank with down sampling and up sampling by 2 is as shown below



The advantage of this method is that the two sub-bands of an input audio or image file can be processed differently and then reconstructed.

**Input as speech signal:**

A speech signal is taken and passed through the filter banks and then reconstructed.

**Matlab code:**

[x1,Fs] = audioread('hi.mp3');

ip=x1(:,1);

figure;

stem(abs(fft(ip)));

title('input audio signal spectrum');

xlabel('n');

ylabel('x[n]');

fpass=1000;

fstop= 1400;

fs=8000;

wpass=(2\*fpass)/fs;

wstop=(2\*fstop)/fs;

wc=(fpass+fstop)\*2\*pi/(2\*fs);

tb=((fstop-fpass)\*2\*pi)/fs;

N=(8\*pi)/tb;

N=ceil(N);

h=zeros(1,N);

for n=1:N

h(n)=sin(wc\*(n-ceil(N/2)))/(pi\*(n-ceil(N/2)));

%low pass impulse response

end

h(ceil(N/2))=wc/pi;

p=1-wc/pi;

for x=1:N

w(x)=0.54-0.46\*(cos(2\*pi\*x/(N-1))); %hamming window function

end

hw0=h.\*w; %after windowing

%% bank 1

ylpf=conv(ip,hw0); %passing through LPF

dn1=downsample(ylpf,2); %downsampled by 2

up1=upsample(dn1,2); %upsampled by 2

y1=conv(up1,hw0); %bank1 lpf out

h2=-h; %HPF impulse response

h2(ceil(N/2))=p;

hw1=h2.\*w; %after windowing

%% bank 2

yhpf=conv(ip,hw1); %passing through HPF

dn2=downsample(yhpf,2); %downsampled by 2

up2=upsample(dn2,2); %upsampled by 2

y2=conv(up2,hw1); %HPF out

res=y1+y2;

figure;

stem(abs(fft(res)));

title('Recovered audio signal spectrum');

xlabel('n');

ylabel('y[n]');

sound(res,Fs);

figure,

subplot(2,2,1),

stem(abs(fft(ylpf)));xlabel('n');ylabel('ylpf[n]');

title('Output of analysis low pass filter');

subplot(2,2,2),

stem(abs(fft(dn1)));xlabel('n');ylabel('dn1[n]');

title('Output of down sampler-low pass');

subplot(2,2,3),

stem(abs(fft(up1)));xlabel('n');ylabel('up1[n]');

title('Output of up sampler-low pass');

subplot(2,2,4),

stem(abs(fft(y1)));xlabel('n');ylabel('y1[n]');

title('Output of synthesis low pass filter');

figure,

subplot(2,2,1),

stem(abs(fft(yhpf)));xlabel('n');ylabel('yhpf[n]');

title('Output of analysis high pass filter');

subplot(2,2,2),

stem(abs(fft(dn2)));xlabel('n');ylabel('dn2[n]');

title('Output of down sampler-high pass');

subplot(2,2,3),

stem(abs(fft(up2)));xlabel('n');ylabel('up2[n]');

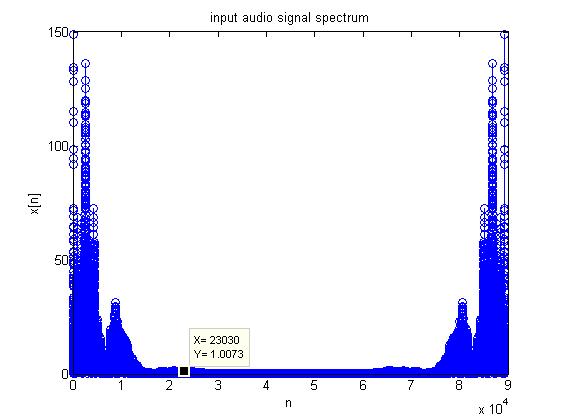
title('Output of up sampler-high pass');

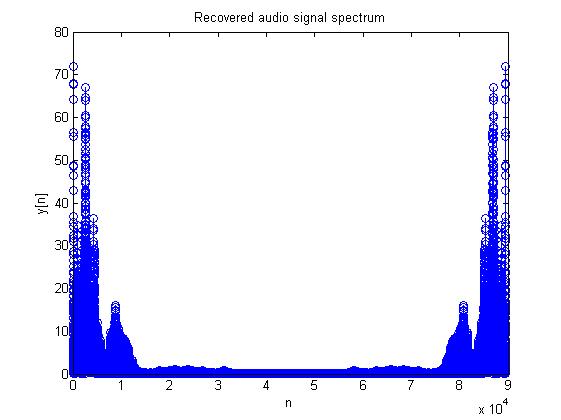
subplot(2,2,4),

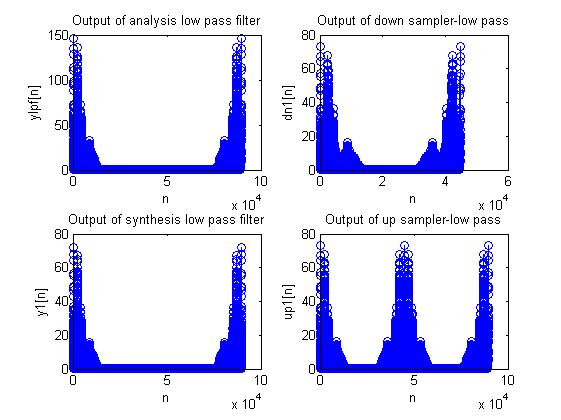
stem(abs(fft(y2)));xlabel('n');ylabel('y2[n]');

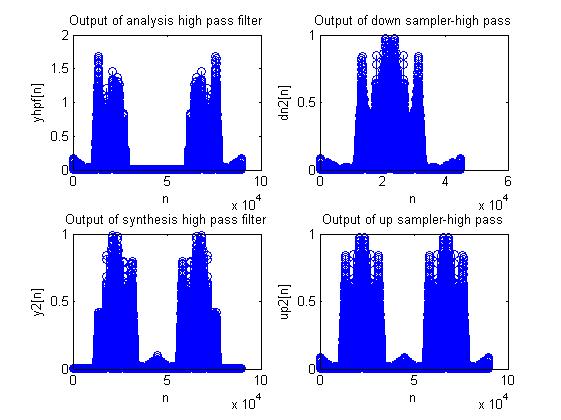
title('Output of synthesis high pass filter');

**Plots:**

****

****

****

****

**Input as color image:**

For color image we will be having 3channels of 2dimensions each (corresponding to Red, Green and Blue components), so we process 3channels separately and then combine all of them to get the final reconstructed image.

**Matlab code:**

clc

clear all;

close all;

f=(imread('peppers.png'));

g=im2double(f);

%%1st channel

img(:,:)=g(:,:,1);

[r,c]=size(img);

order = 50;

n = 0:order;

hlpf = sin(0.875\*pi.\*(n-(order/2)))./(pi.\*(n-(order/2)));

%low pass impulse response

hlpf((order/2)+1) = 0.5;

hlpf2=ftrans2(hlpf); %converting 1-D impulse response to 2-D

hhpf = -hlpf;

%high pass impulse response

hhpf((order/2)+1) = 1 + hhpf((order/2)+1);

hhpf2=ftrans2(hhpf);

ay\_lpf = filter2(hlpf2,img); %analysis low pass

ay\_hpf = filter2(hhpf2,img); %analysis high pass

ay\_lpf\_merge(:,:,1)=ay\_lpf;

ay\_hpf\_merge(:,:,1)=ay\_hpf;

dlpf = ay\_lpf(1:2:r,1:2:c); %downsampled low pass

dhpf = ay\_hpf(1:2:r,1:2:c); %downsampled high pass

dlpf\_merge(:,:,1)=dlpf;

dhpf\_merge(:,:,1)=dhpf;

ulpf = zeros(r,c);

uhpf = zeros(r,c);

for i = 1:r/2

for j = 1:c/2

ulpf(2\*i,2\*j) = dlpf(i,j);

ulpf(2\*i-1,2\*j-1) = dlpf(i,j);

%upsampled low pass

uhpf(2\*i,2\*j) = dhpf(i,j);

uhpf(2\*i-1,2\*j-1) = dhpf(i,j);

%upsampled high pass

end

end

ulpf\_merge(:,:,1)=ulpf;

uhpf\_merge(:,:,1)=uhpf;

sy\_lpf = filter2(hlpf2,ulpf); %synthesis low pass

sy\_hpf = filter2(hhpf2,uhpf); %synthesis high pass

sy\_lpf\_merge(:,:,1)=sy\_lpf;

sy\_hpf\_merge(:,:,1)=sy\_hpf;

final1 = sy\_lpf+sy\_hpf; %adding both the bank outputs

%2nd channel

img(:,:)=g(:,:,2);

[r,c]=size(img);

ay\_lpf = filter2(hlpf2,img);

ay\_hpf = filter2(hhpf2,img);

ay\_lpf\_merge(:,:,2)=ay\_lpf;

ay\_hpf\_merge(:,:,2)=ay\_hpf;

dlpf = ay\_lpf(1:2:r,1:2:c);

dhpf =ay\_hpf(1:2:r,1:2:c);

dlpf\_merge(:,:,2)=dlpf;

dhpf\_merge(:,:,2)=dhpf;

ulpf = zeros(r,c);

uhpf = zeros(r,c);

for i = 1:r/2

for j = 1:c/2

ulpf(2\*i,2\*j) = dlpf(i,j);

ulpf(2\*i-1,2\*j-1) = dlpf(i,j);

uhpf(2\*i,2\*j) = dhpf(i,j);

uhpf(2\*i-1,2\*j-1) = dhpf(i,j);

end

end

ulpf\_merge(:,:,2)=ulpf;

uhpf\_merge(:,:,2)=uhpf;

sy\_lpf = filter2(hlpf2,ulpf);

sy\_hpf = filter2(hhpf2,uhpf);

sy\_lpf\_merge(:,:,2)=sy\_lpf;

sy\_hpf\_merge(:,:,2)=sy\_hpf;

final2 = sy\_lpf+sy\_hpf;

%3rd channel

img(:,:)=g(:,:,3);

[r,c]=size(img);

ay\_lpf = filter2(hlpf2,img);

ay\_hpf = filter2(hhpf2,img);

ay\_lpf\_merge(:,:,3)=ay\_lpf;

ay\_hpf\_merge(:,:,3)=ay\_hpf;

dlpf = ay\_lpf(1:2:r,1:2:c);

dhpf = ay\_hpf(1:2:r,1:2:c);

dlpf\_merge(:,:,3)=dlpf;

dhpf\_merge(:,:,3)=dhpf;

ulpf = zeros(r,c);

uhpf = zeros(r,c);

for i = 1:r/2

for j = 1:c/2

ulpf(2\*i,2\*j) = dlpf(i,j);

ulpf(2\*i-1,2\*j-1) = dlpf(i,j);

uhpf(2\*i,2\*j) = dhpf(i,j);

uhpf(2\*i-1,2\*j-1) = dhpf(i,j);

end

end

ulpf\_merge(:,:,3)=ulpf;

uhpf\_merge(:,:,3)=uhpf;

sy\_lpf = filter2(hlpf2,ulpf);

sy\_hpf = filter2(hhpf2,uhpf);

sy\_lpf\_merge(:,:,3)=sy\_lpf;

sy\_hpf\_merge(:,:,3)=sy\_hpf;

final3 = sy\_lpf+sy\_hpf;

imgc(:,:,1)=final1;

imgc(:,:,2)=final2;

imgc(:,:,3)=final3;

figure, imshow(g);

title('Original image');

figure, imshow(imgc);

title('Reconstructed image');

figure,

subplot(2,2,1),

imshow(ay\_lpf\_merge);title('Output of analysis low pass filter');

subplot(2,2,3),

imshow(sy\_lpf\_merge);title('Output of synthesis low pass filter');

subplot(2,2,2),

imshow(dlpf\_merge);title('Output of down sampler-low pass');

subplot(2,2,4),

imshow(ulpf\_merge);title('Output of up sampler-low pass');

figure,

subplot(2,2,1),

imshow(ay\_hpf\_merge);title('Output of analysis high pass filter');

subplot(2,2,3),

imshow(sy\_hpf\_merge);title('Output of synthesis high pass filter');

subplot(2,2,2),

imshow(dhpf\_merge);title('Output of down sampler-high pass');

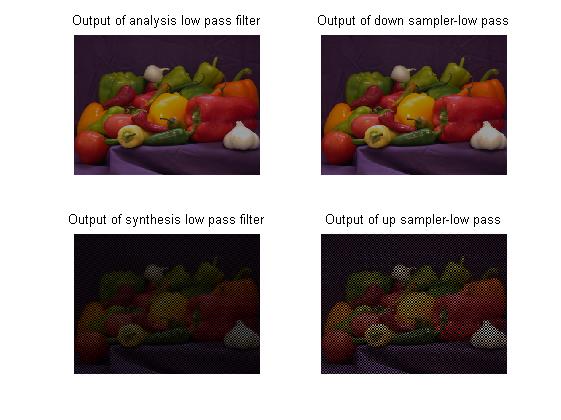
subplot(2,2,4),

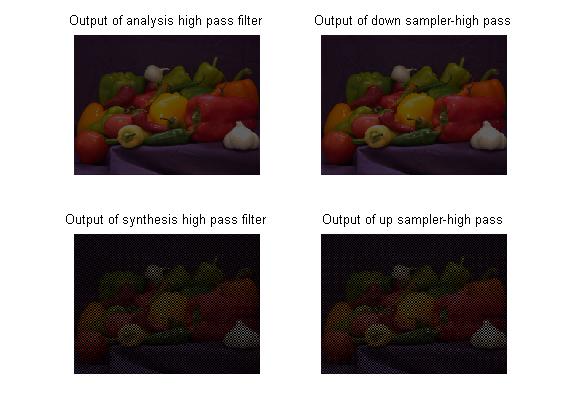
imshow(uhpf\_merge);title('Output of up sampler-high pass');

**Plots:**

****

****

****

****

**Input as black and white image:**

For black and white image, we will have only 1 channel of 2dimensions (corresponding to black and white components). So, we pass the image signal to the filter bank and reconstruct the image.

**Matlab Code:**

clc

clear all;

close all;

f=rgb2gray(imread('lena\_bw1.png'));

g=im2double(f);

img(:,:)=g;

[r,c]=size(img);

order = 50;

n = 0:order;

hlpf = sin(0.875\*pi.\*(n-(order/2)))./(pi.\*(n-(order/2)));

hlpf((order/2)+1) = 0.5;

hlpf2=ftrans2(hlpf);

hhpf = -hlpf;

hhpf((order/2)+1) = 1 + hhpf((order/2)+1);

hhpf2=ftrans2(hhpf);

ay\_lpf = filter2(hlpf2,img);

ay\_hpf = filter2(hhpf2,img);

ay\_lpf\_merge(:,:,1)=ay\_lpf;

ay\_hpf\_merge(:,:,1)=ay\_hpf;

dlpf = ay\_lpf(1:2:r,1:2:c);

dhpf = ay\_hpf(1:2:r,1:2:c);

dlpf\_merge(:,:,1)=dlpf;

dhpf\_merge(:,:,1)=dhpf;

ulpf = zeros(r,c);

uhpf = zeros(r,c);

for i = 1:r/2

for j = 1:c/2

ulpf(2\*i,2\*j) = dlpf(i,j);

ulpf(2\*i-1,2\*j-1) = dlpf(i,j);

uhpf(2\*i,2\*j) = dhpf(i,j);

uhpf(2\*i-1,2\*j-1) =dhpf(i,j);

end

end

sy\_lpf = filter2(hlpf2,ulpf);

sy\_hpf = filter2(hhpf2,uhpf);

final= sy\_lpf+sy\_hpf;

imgc=final;

figure, imshow(g);

title('Original image');

figure, imshow(imgc);

title('Reconstructed image');

figure,

subplot(2,2,1),

imshow(ay\_lpf);title('Output of analysis low pass filter');

subplot(2,2,3),

imshow(sy\_lpf);title('Output of synthesis low pass filter');

subplot(2,2,2),

imshow(dlpf);title('Output of down sampler-low pass');

subplot(2,2,4),

imshow(ulpf);title('Output of up sampler-low pass');

figure,

subplot(2,2,1),

imshow(ay\_hpf);title('Output of analysis high pass filter');

subplot(2,2,3),

imshow(sy\_hpf);title('Output of synthesis high pass filter');

subplot(2,2,2),

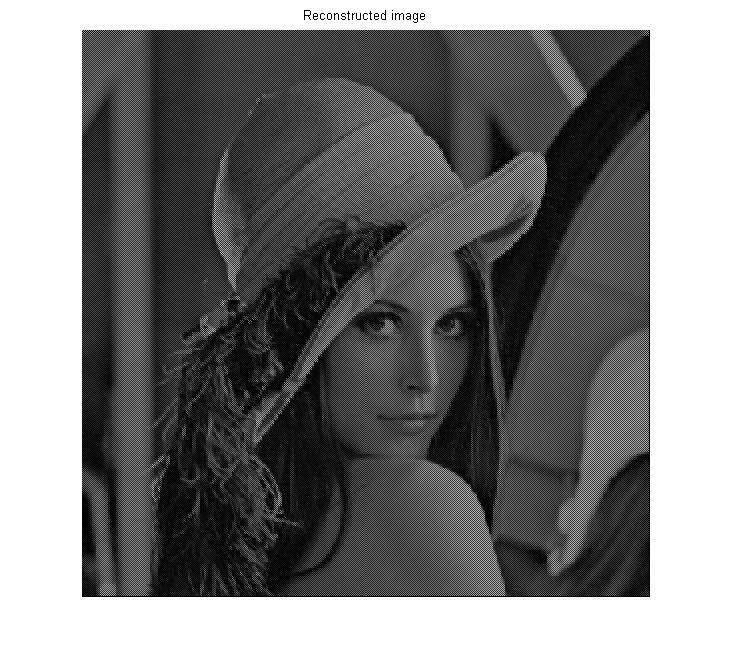
imshow(dhpf);title('Output of down sampler-high pass');

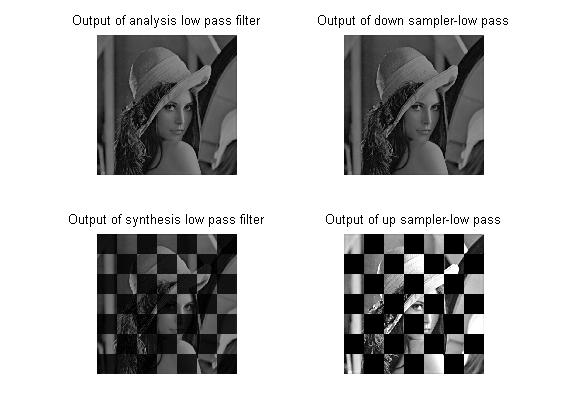
subplot(2,2,4),

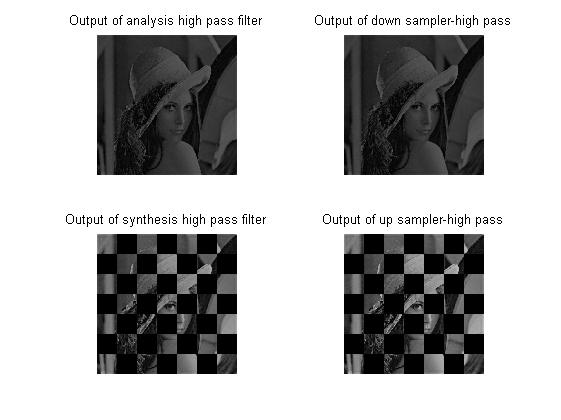
imshow(uhpf);title('Output of up sampler-high pass');

Plots:









**In-built Matlab functions used:**

1. **[y,Fs] = audioread(filename)** reads data from the file named filename, and returns sampled data,y, and a sample rate for that data, Fs.
2. **sound(y,Fs)** sends audio signal y to the speaker at sample rate Fs.
3. **A = imread(filename, fmt)** reads a grayscale or color image from the file specified by the string filename
4. **I2 = im2double(I)** converts the intensity image I to double precision, rescaling the data if necessary.

If the input image is of class double, the output image is identical.

**RGB2 = im2double(RGB)** converts the truecolor image RGB to double

precision, rescaling the data if necessary.

**I = im2double(BW**) converts the binary image BW to a double-precision intensity image.

1. **h = ftrans2(b)** produces the two-dimensional FIR filter h that corresponds to the one-dimensional FIR filter b.
2. **Y = filter2(h,X)** filters the data in X with the two-dimensional FIR filter in the matrix h. It computes the result, Y, using two-dimensional correlation, and returns the central part of the correlation that is the same size as X.
3. **imshow(I)** displays the image I in a figure, where I is a grayscale, RGB (truecolor), or binary image. For binary images, imshow displays pixels with the value 0 (zero) as black and 1 as white.
4. **I = rgb2gray(RGB)** converts the true colour image RGB to the grayscale intensity image I. rgb2gray converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

**Inferences:**

* We observe that the up sampled images in all the cases are having black lines because of loss of data during up sampling.
* Since edges have high frequency we can infer from the figures that edges were notable in high pass images.

**Results:** Thus,the two sub-bands (low pass and high pass) of an input audio or image file can be processed separately and then combined to get the reconstructed signal.