**NATIONAL INSTITUTE OF TECHNOLOGY CALICUT**

**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



**DIGITAL SIGNAL PROCESSING LAB**

**FILTER BANKS REPORT**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted by

Ch.Harika – B150898EC

Ch.Vineesha - B151067EC

Ch.Krishna Teja-B150611EC

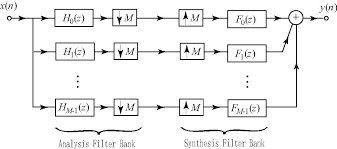
Ch.Sowmya-B150796EC

**AIM:**

To design and implement a filter bank to reconstruct the original audio/image/speech signal.

**THEORY:**

Filter banks, have several applications in communication, speech processing, image processing and compression etc. One of the main requirements in filter bank design is Perfect Reconstruction (PR) which means the signal doesn’t get corrupted by the filter bank. To achieve PR, the output must be a delayed version of the input. Generally, filter banks can be categorized into two main groups: uniform filter bank in which all sampling rates and non-uniform filter bank in which at least one sampling rate is different from the others.

****

A filter bank is a collection of digital filters with a common input or common output. The analysis filter bank is comprised of the M analysis filters Hk(z), which splits the input signal x(n) into M sub-band signals xk(n). The synthesis filter bank uses filters Fk(z) to create a reconstructed output signal xh(n). [Down sampling](http://www.dspdesignline.com/howto/207400771) by M is done just after the analysis bank and [up sampling](http://www.dspdesignline.com/howto/207400771) by M is done before the synthesis bank. The sub-band processing application is usually done in the frequency domain at a slower rate. Designing consists of choosing various filter bank parameters to minimize the error in the reconstructed signal xh(n), while optimizing the performance of the signal decomposition. The analysis part transforms the input signal {\displaystyle x[n]}into M filtered and down sampled outputs {\displaystyle y\_{j}[n],}{\displaystyle j=0,1,...,N-1}. The synthesis part recovers the original signal from {\displaystyle y\_{j}[n]}by up sampling and filtering.

The input signal is decomposed into M so called subband signals by applying M analysis filters with different passbands. Thus, each of the subband signals carries information on the input signal in a particular frequency band. The blocks with arrows pointing downwards in indicate downsampling (subsampling) by factor N, and the blocks with arrows pointing upwards indicate upsampling by N. Subsampling by N means that only every Nth sample is taken. This operation serves to reduce or eliminate redundancies in the N subband signals. Upsampling by N means the insertion of N - 1 consecutive zeros between the samples. This allows us to recover the original sampling rate. The upsamplers are followed by filters which replace the inserted zeros with meaningful values. In the case M =N we speak of critical subsampling, because this is the maximum downsampling factor for which perfect reconstruction can be achieved. Perfect reconstruction means that the output signal is a copy of the input signal with no further distortion than a time shift and amplitude scaling.

**UPSAMPLING:**

 Up sampling can refer to the entire process of increasing the [sampling](https://en.wikipedia.org/wiki/Sampling_(information_theory)) [rate](https://en.wikipedia.org/wiki/Sampling_rate) of a [signal](https://en.wikipedia.org/wiki/Signal_(information_theory)), or it can refer to just one step of the process, the other step being [interpolation](https://en.wikipedia.org/wiki/Interpolation).Rate increase by an integer factor *L* can be explained as a 2-step process, with an equivalent implementation that is more efficient**:**

1. Create a sequence, {\displaystyle \scriptstyle x\_{L}[n],}xL[n] comprising the original samples, {\displaystyle \scriptstyle x[n],}x[n] separated by *L* − 1 zeros. This alone is sometimes referred to as. Up sampling
2. Interpolation: Smooth out the discontinuities with a [lowpass filter](https://en.wikipedia.org/wiki/Lowpass_filter), which replaces the zeros.

**DOWNSAMPLING:**

The process of reducing a sampling rate by an integer factor is referred to as downsampling of a data sequence. We also refer to downsampling as ''decimation''. The term ''decimation'' used for the downsampling process has been accepted and used in many textbooks and fields. To downsample a data sequence x(n) by an integer factor of M, we use the following notation:

y(m) = x(mM)

where y(m) is the down sampled sequence, obtained by taking a sample from the data sequence x(n) for every M samples.

From the Nyquist sampling theorem, it is known that aliasing can occur in the downsampled signal due to the reduced sampling rate. After downsampling by a factor of M, the new sampling period becomes MT, and therefore the new sampling frequency is

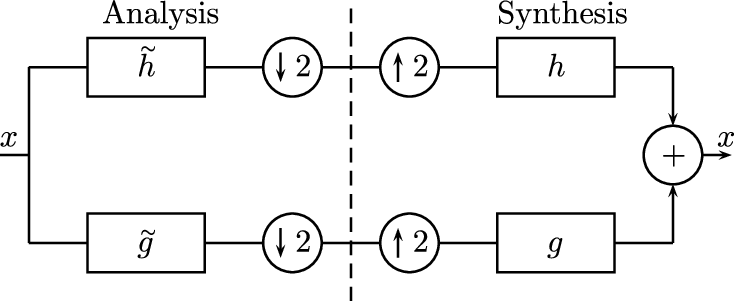
FsM = 1/(MT) = fs / M

where fs is the original sampling rate.

Hence, the folding frequency after downsampling becomes

FsM /2 = fs / (2M)

This tells us that after downsampling by a factor of M, the new folding frequency will be decreased M times. If the signal to be downsampled has frequency components larger than the new folding frequency, f > fs/(2M), aliasing noise will be introduced into the downsampled data.



Two channel filter bank with down and up sampling by 2

**MATLAB COMMANDS USED:**

* [h](file:///C:\Program%20Files\MATLAB\R2018a\help\images\ref\ftrans2.html#d119e51596) = ftrans2([b](file:///C:\\Program%20Files\\MATLAB\\R2018a\\help\\images\\ref\\ftrans2.html" \l "d119e51519),[t](file:///C:\Program%20Files\MATLAB\R2018a\help\images\ref\ftrans2.html#d119e51558)) produces the two-dimensional FIR filter h that corresponds to the one-dimensional FIR filter b using the transform t.
* [[y](file:///C:\\Program%20Files\\MATLAB\\R2018a\\help\\matlab\\ref\\audioread.html" \l "btiabil-1-y),[Fs](file:///C:\Program%20Files\MATLAB\R2018a\help\matlab\ref\audioread.html#btiabil-1-Fs)] = audioread([filename](file:///C:\Program%20Files\MATLAB\R2018a\help\matlab\ref\audioread.html#btiabil-1-filename)) reads data from the file named filename, and returns sampled data, y, and a sample rate for that data, Fs.
* sound([y](file:///C:\\Program%20Files\\MATLAB\\R2018a\\help\\matlab\\ref\\sound.html" \l "btj60o3-1_sep_shared-y),[Fs](file:///C:\Program%20Files\MATLAB\R2018a\help\matlab\ref\sound.html#btj60o3-1_sep_shared-Fs)) sends audio signal y to the speaker at sample rate Fs.
* [A](file:///C:\Program%20Files\MATLAB\R2018a\help\matlab\ref\imread.html#btnczv9-1-A) = imread([filename](file:///C:\Program%20Files\MATLAB\R2018a\help\matlab\ref\imread.html#btnczv9-1-filename)) reads the image from the file specified by filename, inferring the format of the file from its contents. If filename is a multi-image file, thenimread reads the first image in the file.
* imshow([I](file:///C:\Program%20Files\MATLAB\R2018a\help\matlab\ref\imshow.html#bvmnrxi-1-I)) displays the grayscale image I in a figure. imshow optimizes figure, axes, and image object properties for image display.
* [C](file:///C:\Program%20Files\MATLAB\R2018a\help\matlab\ref\conv2.html#bvgtez6-C) = conv2([A](file:///C:\Program%20Files\MATLAB\R2018a\help\matlab\ref\conv2.html#bvgtez6-A),[B](file:///C:\Program%20Files\MATLAB\R2018a\help\matlab\ref\conv2.html#bvgtez6-B)) returns the [two-dimensional convolution](file:///C:\Program%20Files\MATLAB\R2018a\help\matlab\ref\conv2.html#bvgtfv6) of matrices A and B.
* [y](file:///C:\Program%20Files\MATLAB\R2018a\help\signal\ref\downsample.html#d119e34413) = downsample([x](file:///C:\\Program%20Files\\MATLAB\\R2018a\\help\\signal\\ref\\downsample.html" \l "d119e34289),[n](file:///C:\Program%20Files\MATLAB\R2018a\help\signal\ref\downsample.html#d119e34341)) decreases the sample rate of x by keeping the first sample and then every nth sample after the first. If x is a matrix, the function treats each column as a separate sequence.
* [y](file:///C:\Program%20Files\MATLAB\R2018a\help\signal\ref\upsample.html#d119e183884) = upsample([x](file:///C:\\Program%20Files\\MATLAB\\R2018a\\help\\signal\\ref\\upsample.html" \l "d119e183760),[n](file:///C:\Program%20Files\MATLAB\R2018a\help\signal\ref\upsample.html#d119e183812)) increases the sample rate of x by inserting n – 1 zeros between samples. If x is a matrix, the function treats each column as a separate sequence.
* I2 = im2double([I](file:///C:\Program%20Files\MATLAB\R2018a\help\matlab\ref\im2double.html#buhz5vt-1-I)) converts the intensity image I to double precision, rescaling the data if necessary. I can be a grayscale intensity image, a true color image, or a binary image. If the input image is of class double, then the output image is identical.

**MATLAB CODE:**

1. Code for reconstruction of audio / speech signals using filter bank.

[in,Fs] = audioread('male.wav');

in1=in(:,1);

figure;

stem(abs(fft(in1)));

title('input');

xlabel('n');

ylabel('x[n]');

fpass=1000;

fstop= 1500;

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;

h=zeros(1,N);

for n=1:N

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

end

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

p=1-wc/pi;

n=1:N;

for j=1:N

w(j)=0.42659-0.5\*(cos((2\*pi\*j)/(N-1)))+0.08\*(cos((2\*pi\*j)/(N-1)));

end

hw0=h.\*w; %after windowing

lpf1=conv(in1,hw0); %passing through LPF

h2=-h; %HPF impulse response

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

hw1=h2.\*w; %after windowing

hpf1=conv(in1,hw1); %passing through HPF

dlpf=lpf1(1:2:length(lpf1)); %downsampled by 2

dhpf=hpf1(1:2:length(hpf1)); %downsampled by 2

u1lpf=upsample(dlpf,2); %upsampled by 2

u1hpf=upsample(dhpf,2); %upsampled by 2

lpf2=conv(ulpf,hw0); %bank1 lpf out

hpf2=conv(uhpf,hw1); %HPF output

res=lpf2+hpf2;

figure;

stem(abs(fft(res)));

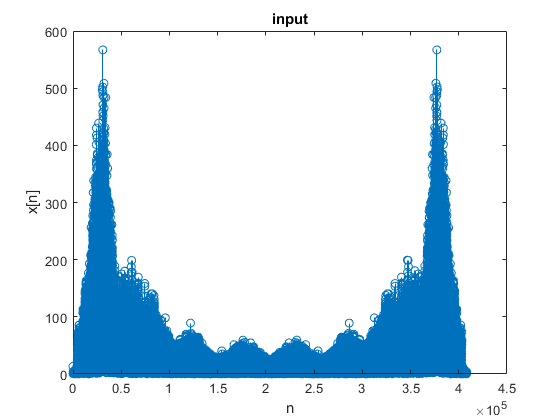
title('Recovered signal');

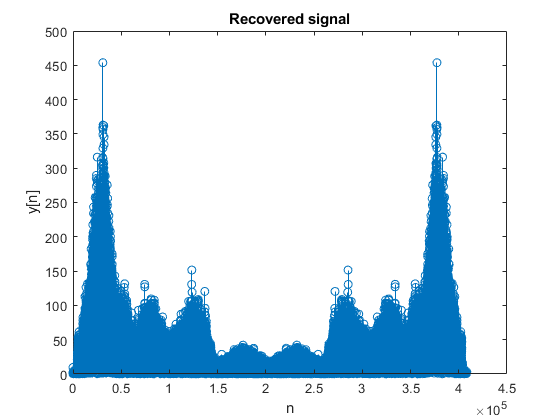
xlabel('n');

ylabel('y[n]');

sound(res,Fs);

OUTPUT:





1. Code for reconstruction of black and white (grayscale) image.

close all;

clear all;

%% lpf

fstop = 1000;

fpass = 500;

fs= 10000;

ws = 2\*pi\*fstop/fs;

wp = 2\*pi\*fpass/fs;

wc = (wp+ws)/2;

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

N= 8\*pi/tb ;

n=[0:N];

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

h((N/2)+1)= wc/pi;

r1 = ftrans2(h);

%% hpf

hh= -h;

hh((N/2)+1) = 1 + hh((N/2)+1);

r2 = ftrans2(hh);

%% input

a=imread('berry.png');

a11= a(: ,: ,1);

a1=im2double(a11);

figure;

imshow(a);

title('input image');

%% stage 1

aalpf = conv2(double(a1),double(r1)); % input through lpf

[x,y] = size(aalpf);

alpf = aalpf(N/2+1: x-N/2,N/2+1:y-N/2);

figure;

subplot(2,1,1)

imshow(alpf);

title('low pass image');

aahpf = conv2(double(a1),double(r2)); % input through hpf

[x1,y1] = size(aahpf);

ahpf = aahpf(N/2+1: x1-N/2,N/2+1:y1-N/2);

subplot(2,1,2)

imshow(ahpf);

title('High pass image');

%% stage 2

downlpf = downsample(alpf,2); % lpf output downsampled

downhpf = downsample(ahpf,2); % lpf output downsampled

figure;

imshow(downlpf);

title('low pass downsampled image');

figure;

imshow(downhpf);

title('High pass up sampled image');

%% stage 3

uplpf = upsample(downlpf,2);

uplpf = 2.\*uplpf;

uphpf = upsample(downhpf,2);

uphpf = 2.\*uphpf;

figure;

imshow(uphpf);

title('High pass downsampled image');

figure;

imshow(uplpf);

title('low pass up sampled image');

%% stage 4

newlpf= conv2(double(uplpf),double(r1));

[x2,y2] = size(newlpf);

finlpf = newlpf(N/2+1: x2-N/2,N/2+1:y2-N/2);

newhpf= conv2(double(uphpf),double(r2));

[x3,y3] = size(newhpf);

finhpf = newhpf(N/2+1: x3-N/2,N/2+1:y3-N/2);

figure;

imshow(newlpf);

title('low pass final image');

figure;

imshow(newhpf);

title('High pass final image');

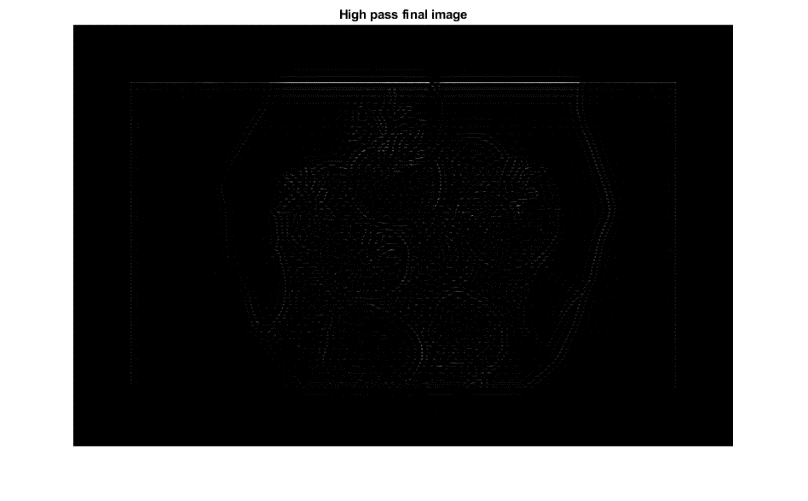
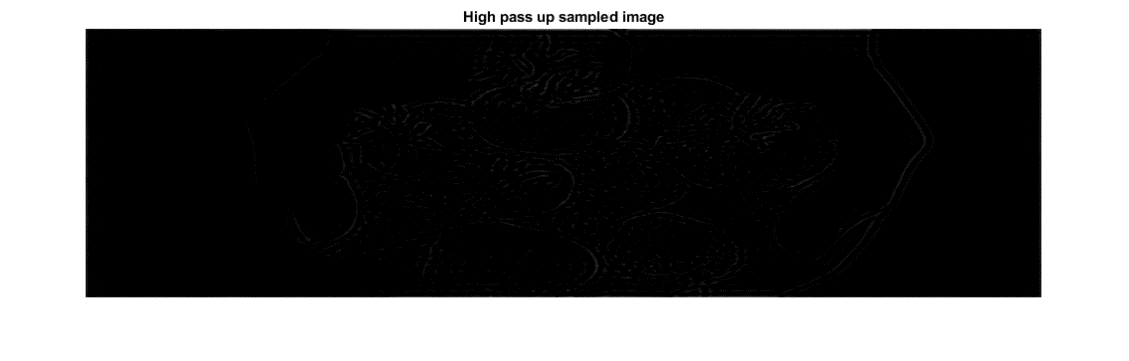
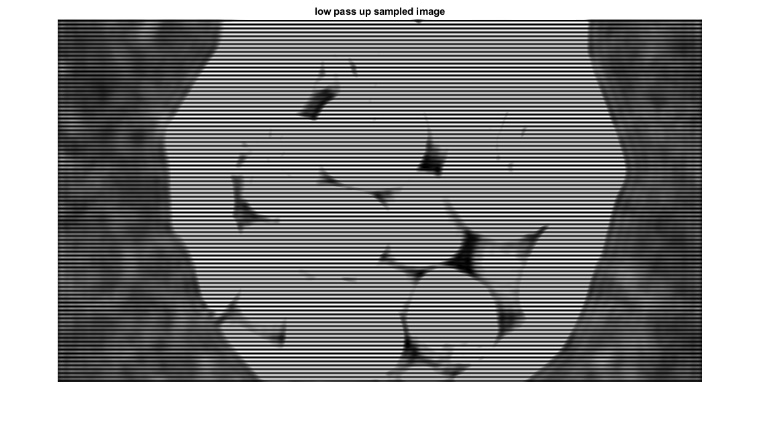
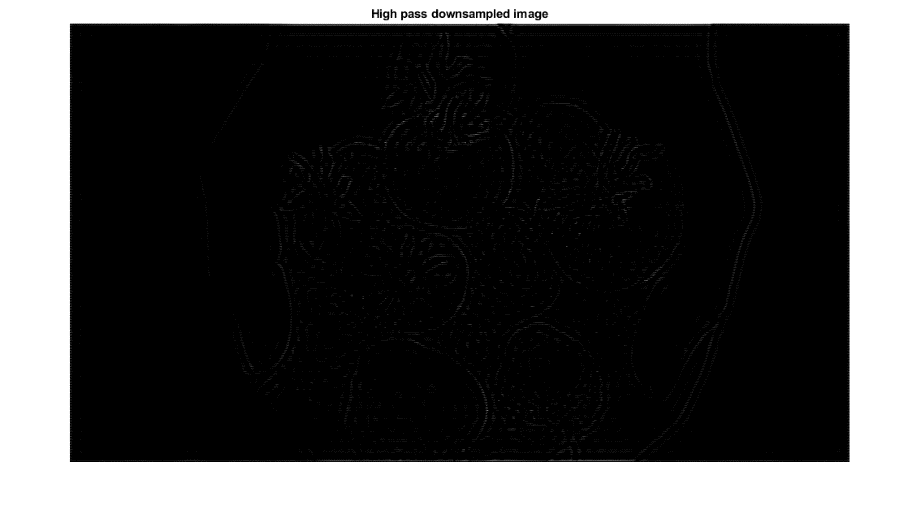
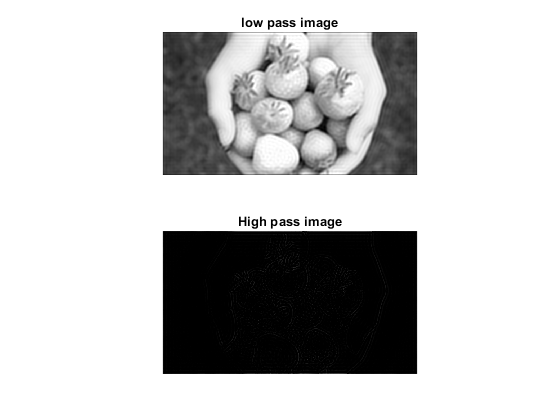
result = finlpf+finhpf;

figure;

imshow(result);

title('reconstructed image');

OUTPUT:





1. Code for reconstruction of color images.

Idea Used:

For processing the color image, all the three dimensional variations must be considered. Hence three 2-D channels were separately processed and combined to obtain the final reconstructed color image.

close all;

clear all;

%% lpf

fstop = 1000;

fpass = 500;

fs= 10000;

ws = 2\*pi\*fstop/fs;

wp = 2\*pi\*fpass/fs;

wc = (wp+ws)/2;

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

N= 8\*pi/tb ;

n=[0:N];

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

h((N/2)+1)= wc/pi;

r1 = ftrans2(h);

%% hpf

hh= -h;

hh((N/2)+1) = 1 + hh((N/2)+1);

r2 = ftrans2(hh);

%% input

a=imread('berry.png');

figure;

imshow(a);

title('input image');

for i = 1:3

a11=im2double(a);

a1(:,:)= a11(: ,: ,i);

%% stage 1

aalpf = conv2(double(a1),double(r1));% input through lpf

[x,y] = size(aalpf);

alpf = aalpf(N/2+1: x-N/2,N/2+1:y-N/2);

aahpf = conv2(double(a1),double(r2)); % input through hpf

[x1,y1] = size(aahpf);

ahpf = aahpf(N/2+1: x1-N/2,N/2+1:y1-N/2);

%% stage 2

downlpf = downsample(alpf,2); % lpf output downsampled

downhpf = downsample(ahpf,2); % lpf output downsampled

%% stage 3

uplpf = upsample(downlpf,2);

uplpf = 4.\*uplpf;

uphpf = upsample(downhpf,2);

uphpf = 4.\*uphpf;

%% stage 4

newlpf= conv2(double(uplpf),double(r1));

[x2,y2] = size(newlpf);

finlpf = newlpf(N/2+1: x2-N/2,N/2+1:y2-N/2);

newhpf= conv2(double(uphpf),double(r2));

[x3,y3] = size(newhpf);

finhpf = newhpf(N/2+1: x3-N/2,N/2+1:y3-N/2);

result = finlpf+finhpf;

final(:,:,i) = result(:,:);

end

figure;

result1=mat2gray(final);

imshow(result1);

title('reconstructed image');

OUTPUT:





**RESULT:**

Hence different signals like audio, image etc. can be passed through filter bank circuit and the filters can be designed so as to retain the original signal.