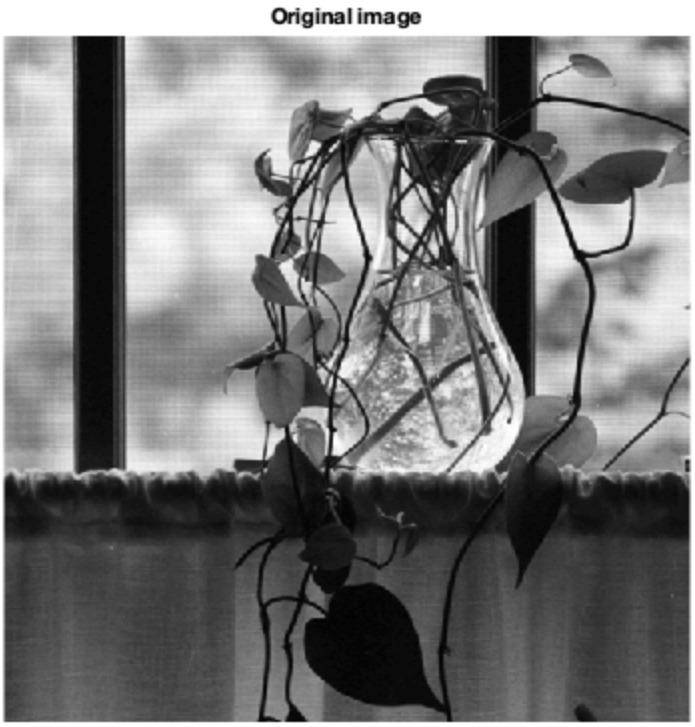
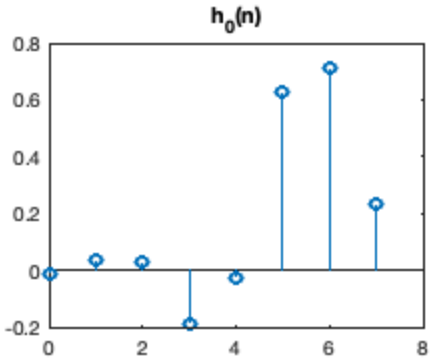
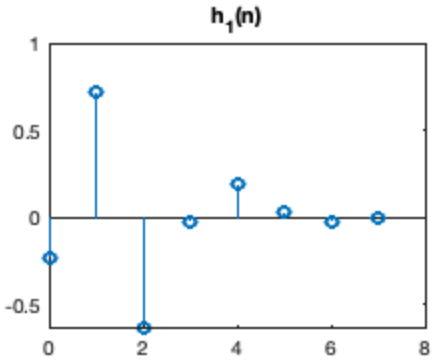
```
% Assignment Submitted By: MANISH SONI
% Problem statement: To use subband filtering twice on the image
%1. first we will demonstrate 512*512 image decomposition to 256*256
 and
%then to construct back the image.
%2. first we will demonstrate 512*512 image decomposition to 128*128
 and
%then to construct back the image.



% part 1
clear all;
close all;

%lets get the original image
original_img = double(imread('plant.tif','tiff'));

figure
imagesc(original_img)
axis('image')
axis('off')
colormap(gray(256))
title('Original image')

[row_len,col_len] = size(original_img);
center = row_len/2 + 1;

%create the filter weights as per the  table 7.1
%   h0 = LFD - low freq decomposition
%   h1 = HFD - high frequency decomposition
%   g0 = LFR - low frequency synthesis
%   g1 = HFR - high frequency synthesis
[LFD,HFD,LFR,HFR] = wfilters('db4');

%lets recreate figure 7.8
num_vec=0:7;
figure
subplot(2,2,1)
stem(num_vec,LFD)
title('h_0(n)')
subplot(2,2,2)
stem(num_vec,HFD)
title('h_1(n)')
subplot(2,2,3)
stem(num_vec,LFR)
title('g_0(n)')
subplot(2,2,4)
stem(num_vec,HFR)
title('g_1(n)')

%apply h0 and h1 to columns of data
```

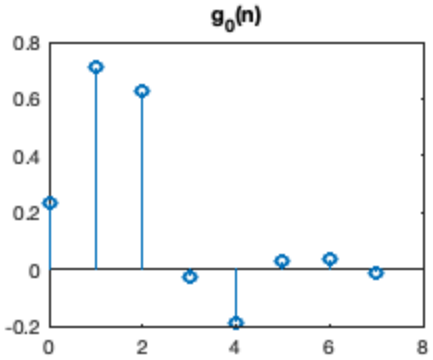
```
h0_row_output = zeros(row_len,col_len);
h1_row_output = zeros(row_len,col_len);
for c=1:col_len

    vec1 = original_img(:,c);  %get the columns sequentially

    %apply hi and low decomposition filters to column data
    h0_col_output(:,c) = conv(vec1,LFD,'same');
    h1_col_output(:,c) = conv(vec1,HFD,'same');

end;

%downsample the columns by a factor of 2 - these images are 256x512
downsampled_h0 = h0_col_output(1:2:row_len,:);
downsampled_h1 = h1_col_output(1:2:row_len,:);

% lets repeat process for rows - note, result generated would be 4
 output arrays

%apply h0 and h1 to row of data for both h0 and h1 outputs

h0_h0_res = zeros(row_len/2,col_len);
h0_h1_res = zeros(row_len/2,col_len);
h1_h0_res = zeros(row_len/2,col_len);
h1_h1_res = zeros(row_len/2,col_len);

for r=1:row_len/2

    vec1 = downsampled_h0(r,:);  %get the columns sequentially
    h0_h0_res(r,:) = conv(vec1,LFD,'same');
    h0_h1_res(r,:) = conv(vec1,HFD,'same');

    vec1 = downsampled_h1(r,:);  %get the columns sequentially
    h1_h0_res(r,:) = conv(vec1,LFD,'same');
    h1_h1_res(r,:) = conv(vec1,HFD,'same');

end;

%downsample the columns by a factor of 2 - these images are 256x512
downsampled_h0_h0 = h0_h0_res(:,1:2:col_len);
downsampled_h0_h1 = h0_h1_res(:,1:2:col_len);
downsampled_h1_h0 = h1_h0_res(:,1:2:col_len);
downsampled_h1_h1 = h1_h1_res(:,1:2:col_len);

%lets display the ouptput
figure
subplot(2,2,1)
imagesc(downsampled_h0_h0)
axis('image')
axis('off')
colormap(gray(256))
title('Approximation image')
subplot(2,2,2)
imagesc(downsampled_h0_h1)
```

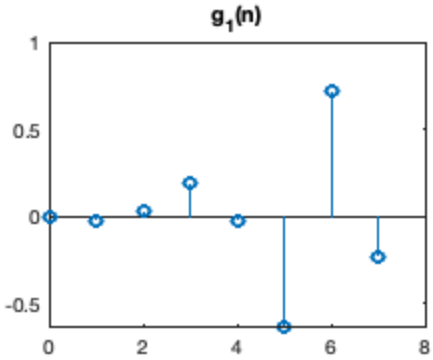
```
axis('image')
axis('off')
colormap(gray(256))
title('Horizontal Detailed Image')
subplot(2,2,3)
imagesc(downsampled_h1_h0)
axis('image')
axis('off')
colormap(gray(256))
title('Vertical Detailed Image')
subplot(2,2,4)
imagesc(downsampled_h1_h1)
axis('image')
axis('off')
colormap(gray(256))
title('Diagonal Detailed Image')
```



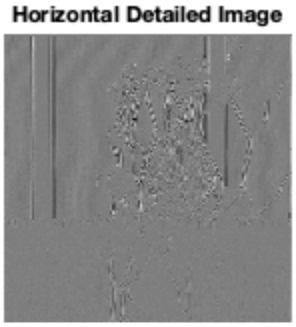

$h_0(n)$
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8

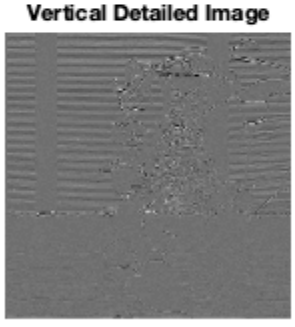

$h_1(n)$
1
0.5
0
-0.5
0
2
4
6
8

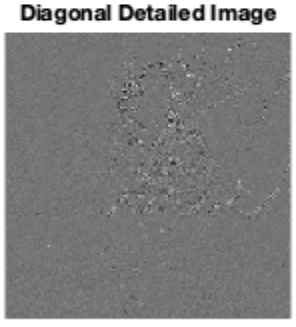

$g_0(n)$
0.8
0.6
0.4
0.2
0
-0.2
0
2
4
6
8


$g_1(n)$
1
0.5
0
-0.5
0
2
4
6
8


Approximation image


Horizontal Detailed Image


Vertical Detailed Image

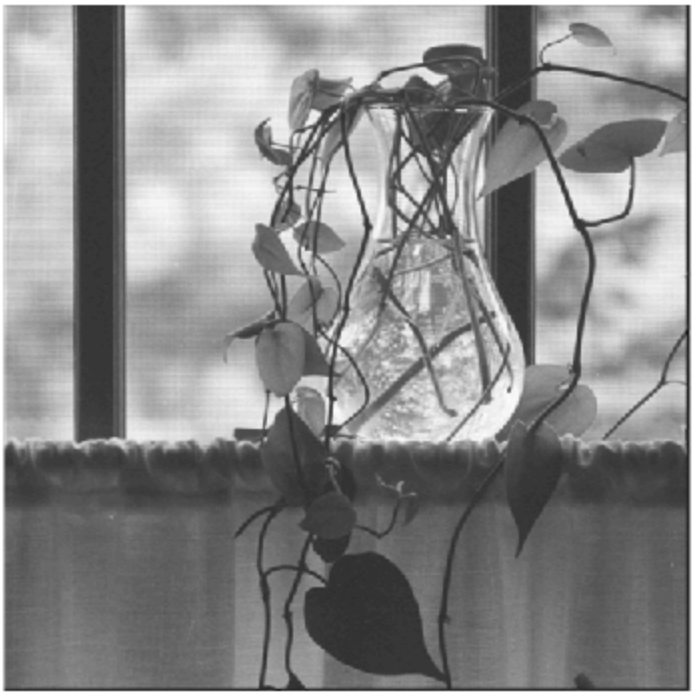

Diagonal Detailed Image

lets reconstruct the image again

```
%apply h0 and h1 to row of data for both h0 and h1 outputs
h0_h0_g0_ip = zeros(row_len/2,col_len);
h0_h1_g1_ip = zeros(row_len/2,col_len);
h1_h0_g0_ip = zeros(row_len/2,col_len);
h1_h1_g1_ip = zeros(row_len/2,col_len);

%upsample columns
for c=1:col_len/2

    h0_h0_g0_ip(:,2*c) = downsampled_h0_h0(:,c);
    h0_h1_g1_ip(:,2*c) = downsampled_h0_h1(:,c);
    h1_h0_g0_ip(:,2*c) = downsampled_h1_h0(:,c);
    h1_h1_g1_ip(:,2*c) = downsampled_h1_h1(:,c);

end;

for r=1:row_len/2

    vec1 = h0_h0_g0_ip(r,:);  %get the columns sequentially
    h0_h0_g0_output(r,:) = conv(vec1,LFR,'same');

    vec1 = h0_h1_g1_ip(r,:);  %get the columns sequentially
    h0_h1_g1_output(r,:) = conv(vec1,HFR,'same');

    vec1 = h1_h0_g0_ip(r,:);  %get the columns sequentially
    h1_h0_g0_output(r,:) = conv(vec1,LFR,'same');

    vec1 = h1_h1_g1_ip(r,:);  %get the columns sequentially
    h1_h1_g1_output(r,:) = conv(vec1,HFR,'same');

end;

%sum
h0_h0_g0_plus_h0_h1_g1 = h0_h0_g0_output + h0_h1_g1_output;
h1_h0_g0_plus_h1_h1_g1 = h1_h0_g0_output + h1_h1_g1_output;

%upsample and process rows
toplayer = zeros(row_len,col_len);
bottomlayer = zeros(row_len,col_len);

%upsample rows
for r=1:row_len/2

    toplayer(2*r,:) = h0_h0_g0_plus_h0_h1_g1(r,:);
    bottomlayer(2*r,:) = h1_h0_g0_plus_h1_h1_g1(r,:);

end;

output_top_layer = zeros(row_len,col_len);
output_bottom_layer = zeros(row_len,col_len);

for c=1:col_len
```

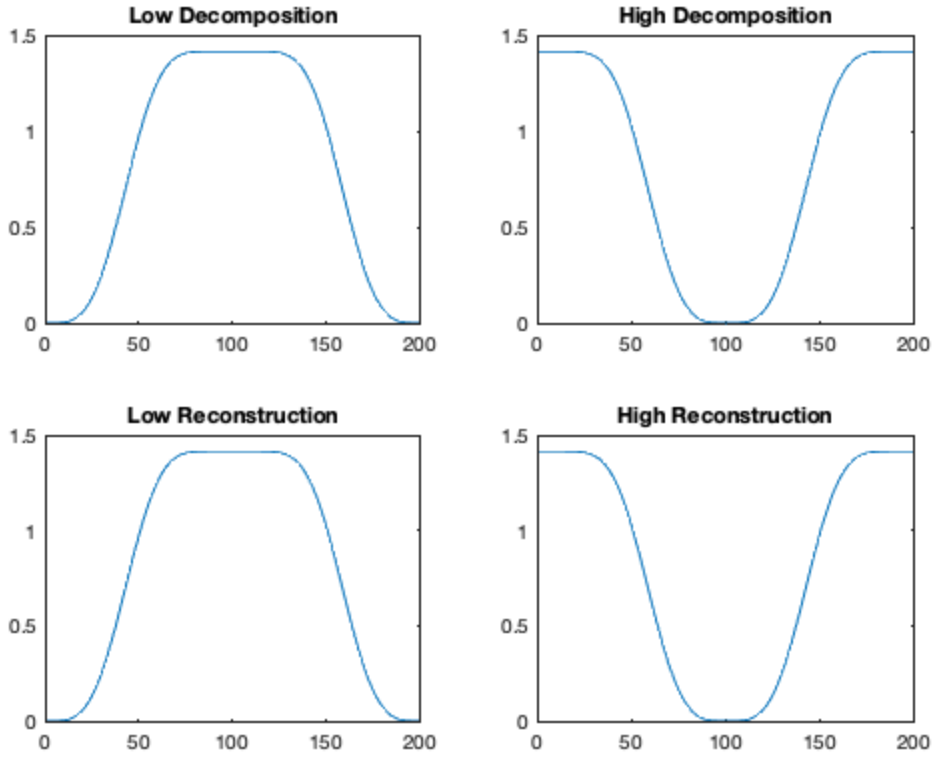
```
    vec1 = toplayer(:,c);
    output_top_layer(:,c) = conv(vec1,LFR,'same');

    vec1 = bottomlayer(:,c);
    output_bottom_layer(:,c) = conv(vec1,HFR,'same');

end;

reconstructed_img = output_top_layer + output_bottom_layer;

reconstructed_img = floor(reconstructed_img*255/
max(max(reconstructed_img)));

figure
imagesc(reconstructed_img)
axis('image')
axis('off')
colormap(gray(256))
title('Reconstructed Image')

difference_imgage = original_img - reconstructed_img;

figure
imagesc(difference_imgage)
axis('image')
axis('off')
colormap(gray(256))
title('input image and reconstructed image difference')

['---------SSIM output of Regenerated image (512*512->256*256-
>512*512) and original--------------------------']
['SSIM Output result for DB4 filter between
 Regenerated image and Original Image =
 ',num2str(100*ssim(reconstructed_img,original_img)),'%']


ans =

    '---------SSIM output of Regenerated image (512*512->256*256-
>512*512) and original--------------------------'


ans =

    'SSIM Output result for DB4 filter between Regenerated image and
 Original Image = 96.4449%'
```

Reconstructed Image



input image and reconstructed image difference

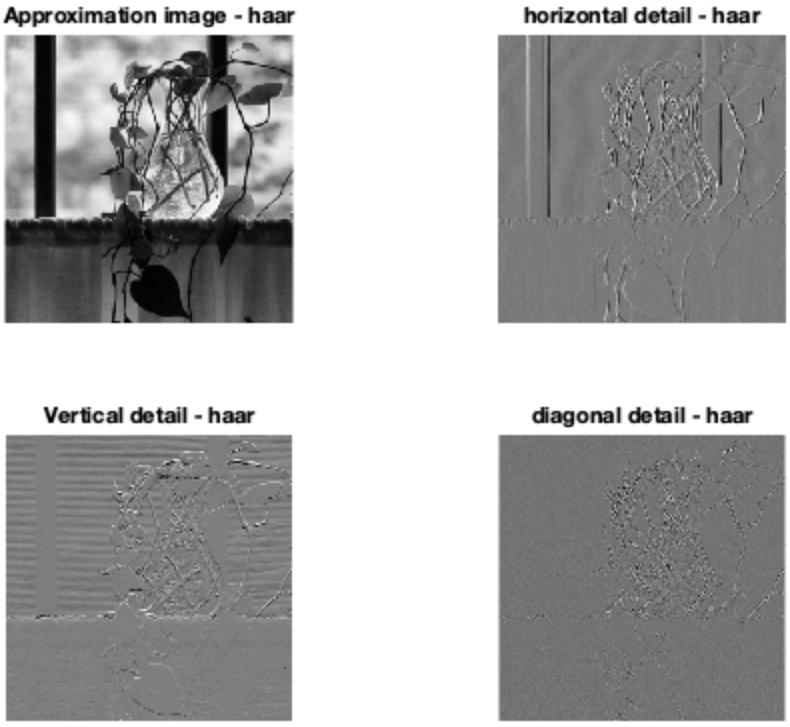
```
LFD_pad = zeros(200,1);
LFD_pad(1:8) = LFD;
ft_LFD = fft(LFD_pad);

HFD_pad = zeros(200,1);
HFD_pad(1:8) = HFD;
ft_HFD = fft(HFD_pad);

LFR_pad = zeros(200,1);
LFR_pad(1:8) = LFR;
ft_LFR = fft(LFR_pad);

HFR_pad = zeros(200,1);
HFR_pad(1:8) = HFR;
ft_HFR = fft(HFR_pad);

figure
subplot(2,2,1)
plot(fftshift(abs(ft_LFD)))
title('Low Decomposition')
subplot(2,2,2)
plot(fftshift(abs(ft_HFD)))
title('High Decomposition')
subplot(2,2,3)
plot(fftshift(abs(ft_LFR)))
title('Low Reconstruction')
subplot(2,2,4)
plot(fftshift(abs(ft_HFR)))
title('High Reconstruction')
```



same example with haar experiment.

```
%haar experiment
[LFD,HFD,LFR,HFR] = wfilters('haar');

%apply h0 and h1 to columns of data
h0_row_output = zeros(row_len,col_len);
h1_row_output = zeros(row_len,col_len);
for c=1:col_len

    vec1 = original_img(:,c);  %get the columns sequentially

    %apply hi and low decomposition filters to column data
    h0_col_output(:,c) = conv(vec1,LFD,'same');
    h1_col_output(:,c) = conv(vec1,HFD,'same');

end;

%downsample the columns by a factor of 2 - these images are 256x512
downsampled_h0 = h0_col_output(1:2:row_len,:);
downsampled_h1 = h1_col_output(1:2:row_len,:);

%repeat process for rows - note, result is 4 output arrays

%apply h0 and h1 to row of data for both h0 and h1 outputs
```

```
h0_h0_res = zeros(row_len/2,col_len);
h0_h1_res = zeros(row_len/2,col_len);
h1_h0_res = zeros(row_len/2,col_len);
h1_h1_res = zeros(row_len/2,col_len);

for r=1:row_len/2

    vec1 = downsampled_h0(r,:);  %get the columns sequentially
    h0_h0_res(r,:) = conv(vec1,LFD,'same');
    h0_h1_res(r,:) = conv(vec1,HFD,'same');

    vec1 = downsampled_h1(r,:);  %get the columns sequentially
    h1_h0_res(r,:) = conv(vec1,LFD,'same');
    h1_h1_res(r,:) = conv(vec1,HFD,'same');

end;

%downsample the columns by a factor of 2 - these images are 256x512
downsampled_h0_h0 = h0_h0_res(:,1:2:col_len);
downsampled_h0_h1 = h0_h1_res(:,1:2:col_len);
downsampled_h1_h0 = h1_h0_res(:,1:2:col_len);
downsampled_h1_h1 = h1_h1_res(:,1:2:col_len);

%display
figure
subplot(2,2,1)
imagesc(downsampled_h0_h0)
axis('image')
axis('off')
colormap(gray(256))
title('Approximation image - haar')
subplot(2,2,2)
imagesc(downsampled_h0_h1)
axis('image')
axis('off')
colormap(gray(256))
title('horizontal detail - haar')
subplot(2,2,3)
imagesc(downsampled_h1_h0)
axis('image')
axis('off')
colormap(gray(256))
title('Vertical detail - haar')
subplot(2,2,4)
imagesc(downsampled_h1_h1)
axis('image')
axis('off')
colormap(gray(256))
title('diagonal detail - haar')
```

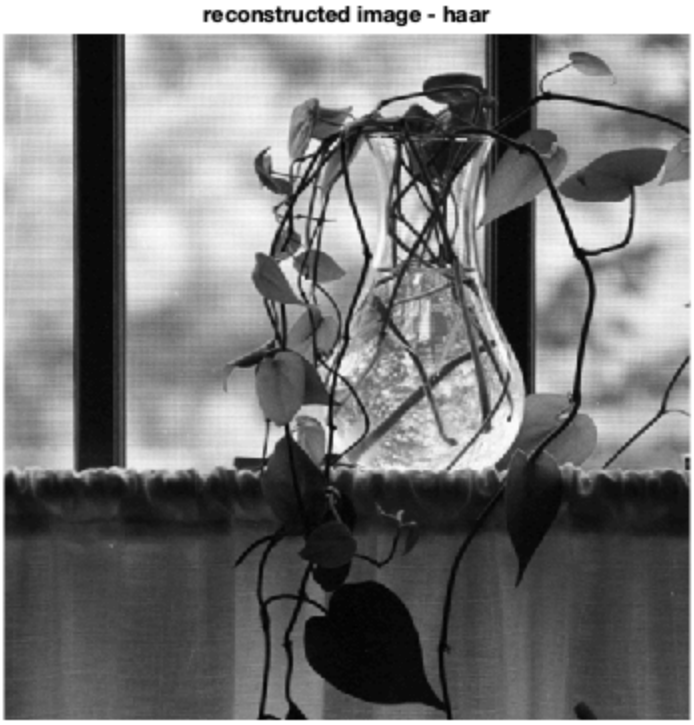


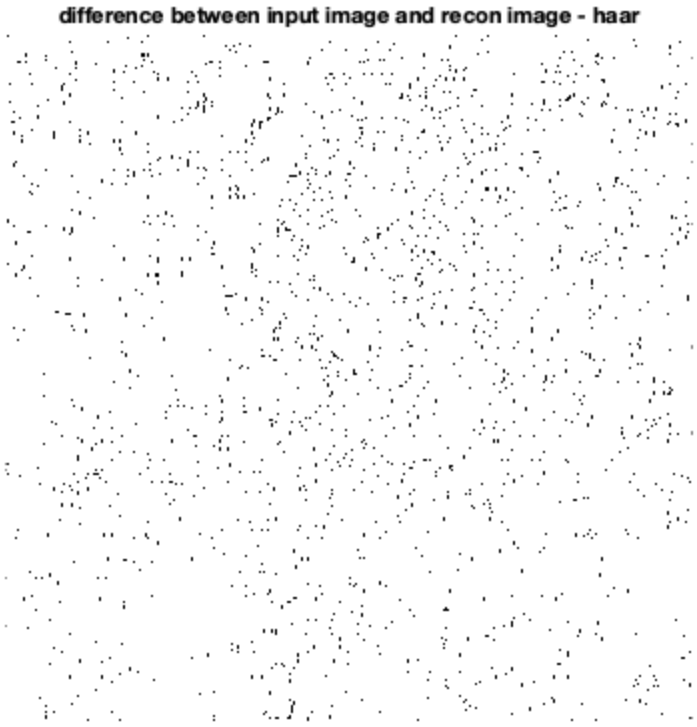
lets demonstrate reconstruction filter - a mirror of Fig 7.7 with g's exchanged

```
%for h's and upsampling instead of downsampling

%apply h0 and h1 to row of data for both h0 and h1 outputs
h0_h0_g0_ip = zeros(row_len/2,col_len);
h0_h1_g1_ip = zeros(row_len/2,col_len);
h1_h0_g0_ip = zeros(row_len/2,col_len);
h1_h1_g1_ip = zeros(row_len/2,col_len);

%upsample columns
for c=1:col_len/2

    h0_h0_g0_ip(:,2*c) = downsampled_h0_h0(:,c);
    h0_h1_g1_ip(:,2*c) = downsampled_h0_h1(:,c);
    h1_h0_g0_ip(:,2*c) = downsampled_h1_h0(:,c);
    h1_h1_g1_ip(:,2*c) = downsampled_h1_h1(:,c);

end;

for r=1:row_len/2

    vec1 = h0_h0_g0_ip(r,:);  %get the columns sequentially
    h0_h0_g0_output(r,:) = conv(vec1,LFR,'same');

    vec1 = h0_h1_g1_ip(r,:);  %get the columns sequentially
```

```
    h0_h1_g1_output(r,:) = conv(vec1,HFR,'same');

    vec1 = h1_h0_g0_ip(r,:);  %get the columns sequentially
    h1_h0_g0_output(r,:) = conv(vec1,LFR,'same');

    vec1 = h1_h1_g1_ip(r,:);  %get the columns sequentially
    h1_h1_g1_output(r,:) = conv(vec1,HFR,'same');

end;

%lets sum up
h0_h0_g0_plus_h0_h1_g1 = h0_h0_g0_output + h0_h1_g1_output;
h1_h0_g0_plus_h1_h1_g1 = h1_h0_g0_output + h1_h1_g1_output;

%upsample and process rows
toplayer = zeros(row_len,col_len);
bottomlayer = zeros(row_len,col_len);

%upsample rows
for r=1:row_len/2

    toplayer(2*r,:) = h0_h0_g0_plus_h0_h1_g1(r,:);
    bottomlayer(2*r,:) = h1_h0_g0_plus_h1_h1_g1(r,:);

end;

output_top_layer = zeros(row_len,col_len);
output_bottom_layer = zeros(row_len,col_len);

for c=1:col_len

    vec1 = toplayer(:,c);
    output_top_layer(:,c) = conv(vec1,LFR,'same');

    vec1 = bottomlayer(:,c);
    output_bottom_layer(:,c) = conv(vec1,HFR,'same');

end;

reconstructed_img = output_top_layer + output_bottom_layer;

reconstructed_img = floor(reconstructed_img*255/
max(max(reconstructed_img)));

figure
imagesc(reconstructed_img)
axis('image')
axis('off')
colormap(gray(256))
title('reconstructed image - haar')

difference_imgage = original_img - reconstructed_img;

figure
```

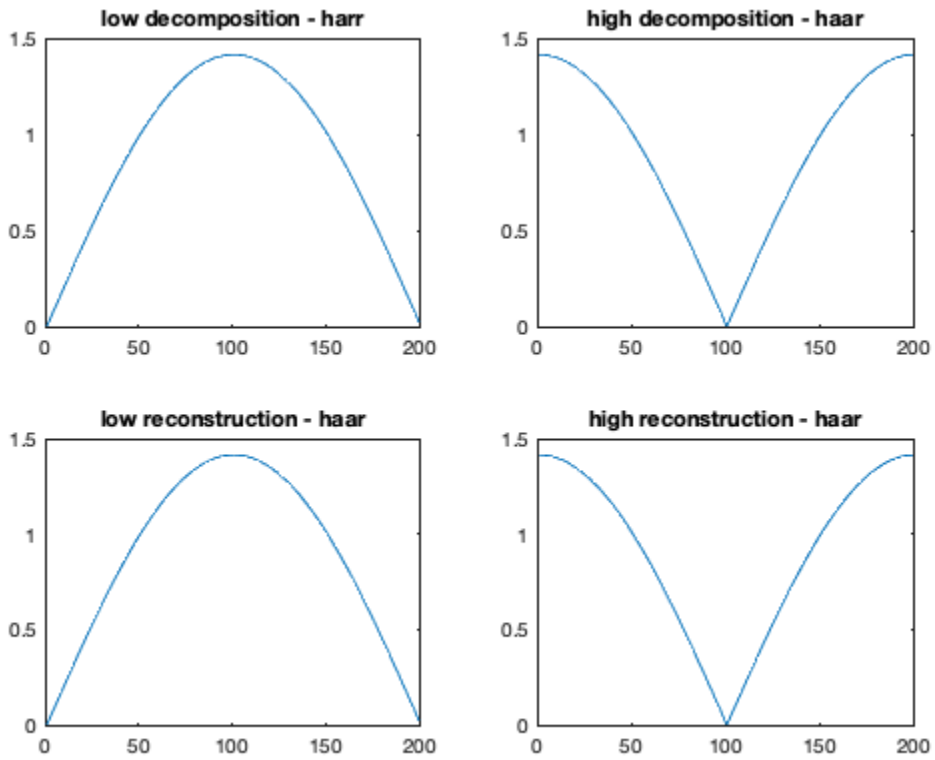
```
imagesc(difference_imgage)
axis('image')
axis('off')
colormap(gray(256))
title('difference between input image and recon image - haar')


['---------SSIM output of Regenerated image (512*512->256*256-
>512*512) and original-------------------------']
['SSIM Output result for Haar filter between
 Regenerated image and Original Image =
 ',num2str(100*ssim(reconstructed_img,original_img)),'%']


ans =

    '---------SSIM output of Regenerated image (512*512->256*256-
>512*512) and original-------------------------'


ans =

    'SSIM Output result for Haar filter between Regenerated image and
 Original Image = 99.9502%'
```






```
LFD_pad = zeros(200,1);
LFD_pad(1:2) = LFD;
ft_LFD = fft(LFD_pad);

HFD_pad = zeros(200,1);
HFD_pad(1:2) = HFD;
ft_HFD = fft(HFD_pad);

LFR_pad = zeros(200,1);
LFR_pad(1:2) = LFR;
ft_LFR = fft(LFR_pad);

HFR_pad = zeros(200,1);
HFR_pad(1:2) = HFR;
ft_HFR = fft(HFR_pad);

figure
subplot(2,2,1)
plot(fftshift(abs(ft_LFD)))
title('low decomposition - harr')
subplot(2,2,2)
plot(fftshift(abs(ft_HFD)))
title('high decomposition - haar')
subplot(2,2,3)
plot(fftshift(abs(ft_LFR)))
title('low reconstruction - haar')
```

```
subplot(2,2,4)
plot(fftshift(abs(ft_HFR)))
title('high reconstruction - haar')
```



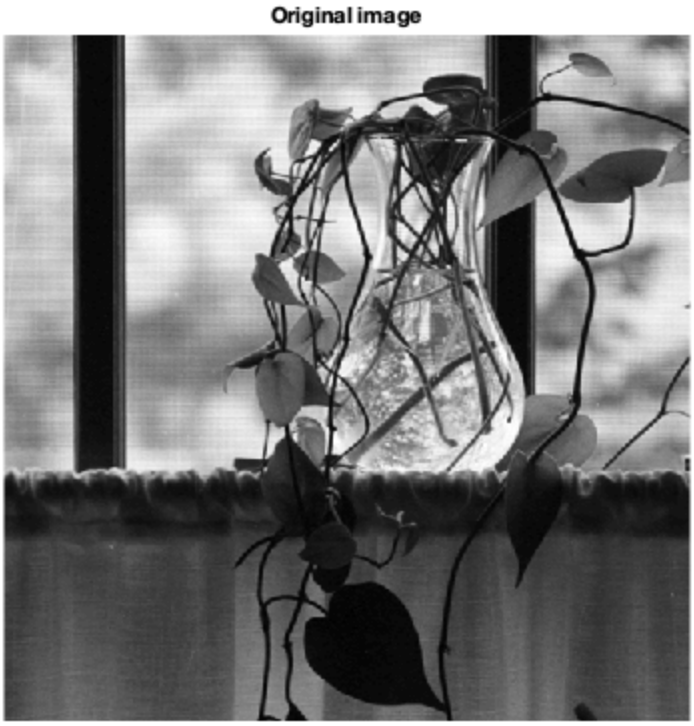
part 2: lets start with original image again this time lets decompose the 512*512 image to 256*256 and then to 128*128 image.

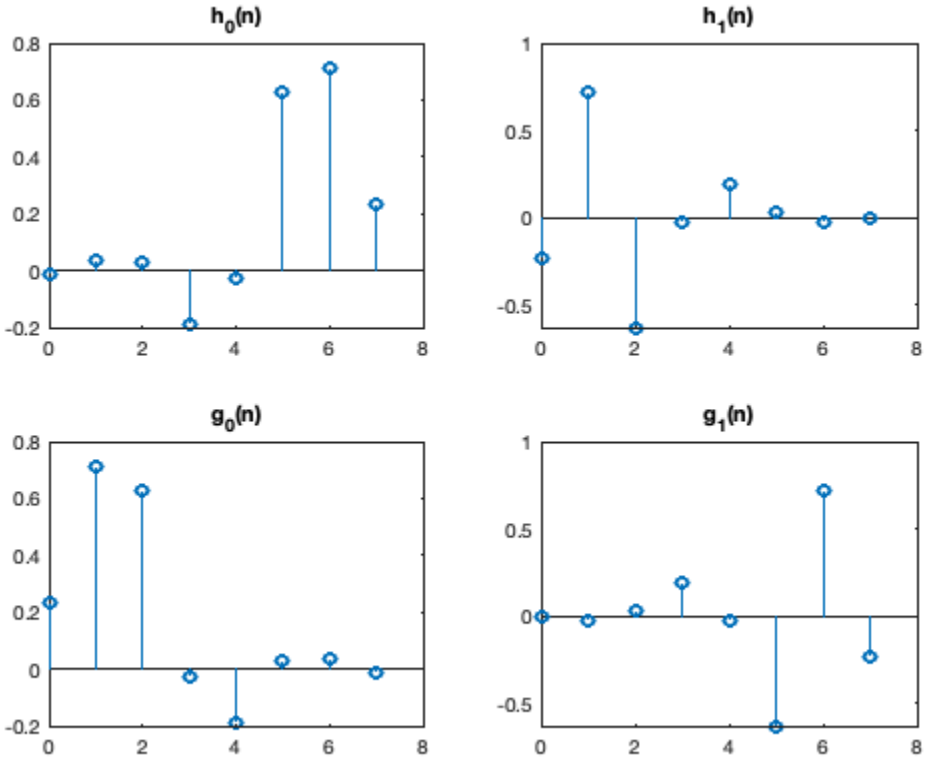
```
original_img = double(imread('plant.tif','tiff'));

figure
imagesc(original_img)
axis('image')
axis('off')
colormap(gray(256))
title('Original image')

[row_len,col_len] = size(original_img);
center = row_len/2 + 1;

%lets create the filter weights as pet the  table 7.1
%h0 = LFD - Low frequency decomposition
%h1 = HFD - High frequency decomposition
%g0 = LFR - Low frequency synthesis
%g1 = HFR - High frequency synthesis
[LFD,HFD,LFR,HFR] = wfilters('db4');

%lets recreate figure 7.8
```

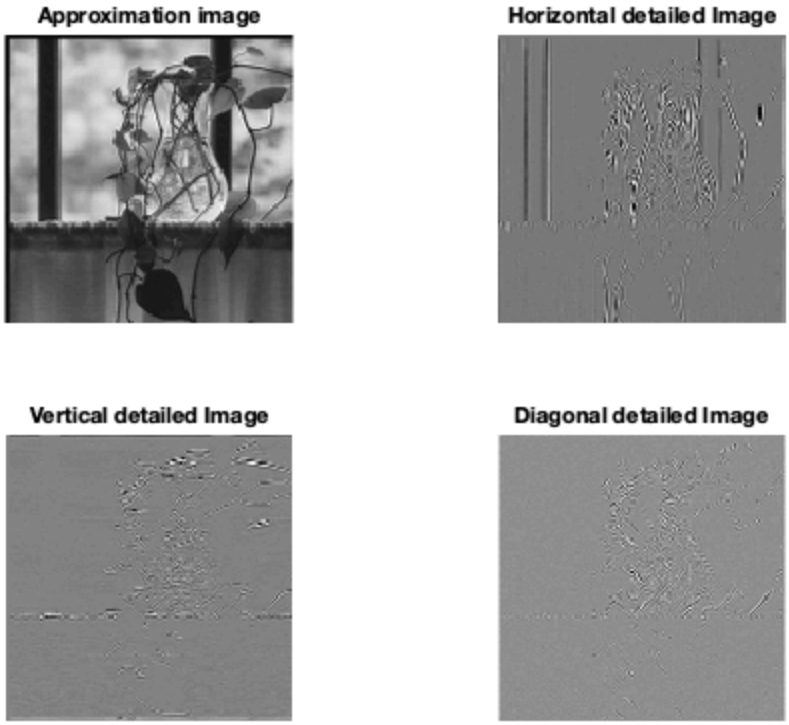
```
num_vec=0:7;
figure
subplot(2,2,1)
stem(num_vec,LFD)
title('h_0(n)')
subplot(2,2,2)
stem(num_vec,HFD)
title('h_1(n)')
subplot(2,2,3)
stem(num_vec,LFR)
title('g_0(n)')
subplot(2,2,4)
stem(num_vec,HFR)
title('g_1(n)')

%apply h0 and h1 to columns of data
h0_row_output = zeros(row_len,col_len);
h1_row_output = zeros(row_len,col_len);
for c=1:col_len

    vec1 = original_img(:,c);  %get the columns sequentially

    % lets apply hi and low decomposition filters to column data
    h0_col_output(:,c) = conv(vec1,LFD,'same');
    h1_col_output(:,c) = conv(vec1,HFD,'same');

end;

%lets downsample the columns by a factor of 2 - these images are
 256x512
downsampled_h0 = h0_col_output(1:2:row_len,:);
downsampled_h1 = h1_col_output(1:2:row_len,:);

%lets repeat process for rows - note, result is 4 output arrays

%lets apply h0 and h1 to row of data for both h0 and h1 outputs

h0_h0_res = zeros(row_len/2,col_len);
h0_h1_res = zeros(row_len/2,col_len);
h1_h0_res = zeros(row_len/2,col_len);
h1_h1_res = zeros(row_len/2,col_len);

for r=1:row_len/2

    vec1 = downsampled_h0(r,:);  %get the columns sequentially
    h0_h0_res(r,:) = conv(vec1,LFD,'same');
    h0_h1_res(r,:) = conv(vec1,HFD,'same');

    vec1 = downsampled_h1(r,:);  %get the columns sequentially
    h1_h0_res(r,:) = conv(vec1,LFD,'same');
    h1_h1_res(r,:) = conv(vec1,HFD,'same');

end;
```

```
%downsample the columns by a factor of 2 - these images are 256x512 to
 get
%256*256 image
downsampled_h0_h0 = h0_h0_res(:,1:2:col_len);
downsampled_h0_h1 = h0_h1_res(:,1:2:col_len);
downsampled_h1_h0 = h1_h0_res(:,1:2:col_len);
downsampled_h1_h1 = h1_h1_res(:,1:2:col_len);


% lets use 256*256 image to decompose it to further 128*128 image

input_img_256_256=downsampled_h0_h0;


[row_len_1,col_len_1] = size(input_img_256_256);

for c=1:col_len_1

    vec1 = input_img_256_256(:,c);  %get the columns sequentially

    %apply hi and low decomposition filters to column data
    h0_col_output_1(:,c) = conv(vec1,LFD,'same');
    h1_col_output_1(:,c) = conv(vec1,HFD,'same');

end;

% lets downsample the columns by a factor of 2 - these images are
 128x256
ds_h0_1 = h0_col_output_1(1:2:row_len_1,:);
ds_h1_1 = h1_col_output_1(1:2:row_len_1,:);

%repeat process for rows - note, result is 4 output arrays

%apply h0 and h1 to row of data for both h0 and h1 outputs

h0_h0_res_1 = zeros(row_len_1/2,col_len_1);
h0_h1_res_1 = zeros(row_len_1/2,col_len_1);
h1_h0_res_1 = zeros(row_len_1/2,col_len_1);
h1_h1_res_1 = zeros(row_len_1/2,col_len_1);

for r=1:row_len_1/2

    vec1 = ds_h0_1(r,:);  %get the columns sequentially
    h0_h0_res_1(r,:) = conv(vec1,LFD,'same');
    h0_h1_res_1(r,:) = conv(vec1,HFD,'same');

    vec1 = ds_h1_1(r,:);  %get the columns sequentially
    h1_h0_res_1(r,:) = conv(vec1,LFD,'same');
    h1_h1_res_1(r,:) = conv(vec1,HFD,'same');

end;

%downsample the columns by a factor of 2 - these images are 128x256 to
 get
```

```
%128*128 image
downsampled_h0_h0_1 = h0_h0_res_1(:,1:2:col_len_1);
downsampled_h0_h1_1 = h0_h1_res_1(:,1:2:col_len_1);
downsampled_h1_h0_1 = h1_h0_res_1(:,1:2:col_len_1);
downsampled_h1_h1_1 = h1_h1_res_1(:,1:2:col_len_1);


%lets display Output
figure
subplot(2,2,1)
imagesc(downsampled_h0_h0_1)
axis('image')
axis('off')
colormap(gray(256))
title('Approximation image')
subplot(2,2,2)
imagesc(downsampled_h0_h1_1)
axis('image')
axis('off')
colormap(gray(256))
title('Horizontal detailed Image')
subplot(2,2,3)
imagesc(downsampled_h1_h0_1)
axis('image')
axis('off')
colormap(gray(256))
title('Vertical detailed Image')
subplot(2,2,4)
imagesc(downsampled_h1_h1_1)
axis('image')
axis('off')
colormap(gray(256))
title('Diagonal detailed Image')
```


Original image


$h_0(n)$
$h_1(n)$
$g_0(n)$
$g_1(n)$



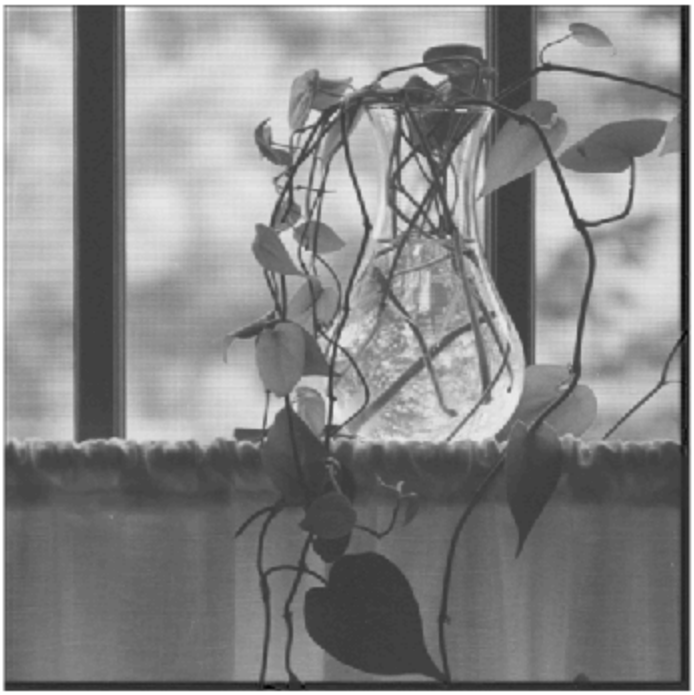

```
% lets start reconstruction of the image using 128*128 image from its
% decomposed components
%lets apply h0 and h1 to row of data for both h0 and h1 outputs

h0_h0_g0_input_1 = zeros(row_len_1/2,col_len_1);
h0_h1_g1_input_1 = zeros(row_len_1/2,col_len_1);
h1_h0_g0_input_1 = zeros(row_len_1/2,col_len_1);
h1_h1_g1_input_1 = zeros(row_len_1/2,col_len_1);

%lets upsample columns
for c=1:col_len_1/2

    h0_h0_g0_input_1(:,2*c) = downsampled_h0_h0_1(:,c);
    h0_h1_g1_input_1(:,2*c) = downsampled_h0_h1_1(:,c);
    h1_h0_g0_input_1(:,2*c) = downsampled_h1_h0_1(:,c);
    h1_h1_g1_input_1(:,2*c) = downsampled_h1_h1_1(:,c);

end;

for r=1:row_len_1/2

    vec1 = h0_h0_g0_input_1(r,:);  %get the columns sequentially
    h0_h0_g0_output_1(r,:) = conv(vec1,LFR,'same');

    vec1 = h0_h1_g1_input_1(r,:);  %get the columns sequentially
    h0_h1_g1_output_1(r,:) = conv(vec1,HFR,'same');
```

```
    vec1 = h1_h0_g0_input_1(r,:);  %get the columns sequentially
    h1_h0_g0_output_1(r,:) = conv(vec1,LFR,'same');

    vec1 = h1_h1_g1_input_1(r,:);  %get the columns sequentially
    h1_h1_g1_output_1(r,:) = conv(vec1,HFR,'same');

end;

%sum
h0_h0_g0_plus_h0_h1_g1_1 = h0_h0_g0_output_1 + h0_h1_g1_output_1;
h1_h0_g0_plus_h1_h1_g1_1 = h1_h0_g0_output_1 + h1_h1_g1_output_1;

% lets do upsampling and process rows
top_1 = zeros(row_len_1,col_len_1);
bottom_1 = zeros(row_len_1,col_len_1);

%lets upsample rows
for r=1:row_len_1/2

    top_1(2*r,:) = h0_h0_g0_plus_h0_h1_g1_1(r,:);
    bottom_1(2*r,:) = h1_h0_g0_plus_h1_h1_g1_1(r,:);

end;

output_top_1 = zeros(row_len_1,col_len_1);
output_bottom_1 = zeros(row_len_1,col_len_1);

for c=1:col_len_1

    vec1 = top_1(:,c);
    output_top_1(:,c) = conv(vec1,LFR,'same');

    vec1 = bottom_1(:,c);
    output_bottom_1(:,c) = conv(vec1,HFR,'same');

end;

recon_img_1 = output_top_1 + output_bottom_1;

%recon_img_1 is a 256*256 image reconsturction. lets use previous
%componenet to get the 512*512 image from 256*256 image

h0_h0_g0_ip = zeros(row_len/2,col_len);
h0_h1_g1_ip = zeros(row_len/2,col_len);
h1_h0_g0_ip = zeros(row_len/2,col_len);
h1_h1_g1_ip = zeros(row_len/2,col_len);

%lets upsample columns
for c=1:col_len/2

    h0_h0_g0_ip(:,2*c) = recon_img_1(:,c);
    h0_h1_g1_ip(:,2*c) = downsampled_h0_h1(:,c);
    h1_h0_g0_ip(:,2*c) = downsampled_h1_h0(:,c);
```

```
    h1_h1_g1_ip(:,2*c) = downsampled_h1_h1(:,c);

end;

for r=1:row_len/2

    vec1 = h0_h0_g0_ip(r,:);  %get the columns sequentially
    h0_h0_g0_output(r,:) = conv(vec1,LFR,'same');

    vec1 = h0_h1_g1_ip(r,:);  %get the columns sequentially
    h0_h1_g1_output(r,:) = conv(vec1,HFR,'same');

    vec1 = h1_h0_g0_ip(r,:);  %get the columns sequentially
    h1_h0_g0_output(r,:) = conv(vec1,LFR,'same');

    vec1 = h1_h1_g1_ip(r,:);  %get the columns sequentially
    h1_h1_g1_output(r,:) = conv(vec1,HFR,'same');

end;

%sum
h0_h0_g0_plus_h0_h1_g1 = h0_h0_g0_output + h0_h1_g1_output;
h1_h0_g0_plus_h1_h1_g1 = h1_h0_g0_output + h1_h1_g1_output;

%upsample and process rows
toplayer = zeros(row_len,col_len);
bottomlayer = zeros(row_len,col_len);

%upsample rows
for r=1:row_len/2

    toplayer(2*r,:) = h0_h0_g0_plus_h0_h1_g1(r,:);
    bottomlayer(2*r,:) = h1_h0_g0_plus_h1_h1_g1(r,:);

end;

output_top_layer = zeros(row_len,col_len);
output_bottom_layer = zeros(row_len,col_len);

for c=1:col_len

    vec1 = toplayer(:,c);
    output_top_layer(:,c) = conv(vec1,LFR,'same');

    vec1 = bottomlayer(:,c);
    output_bottom_layer(:,c) = conv(vec1,HFR,'same');

end;

reconstructed_img = output_top_layer + output_bottom_layer;

reconstructed_img = floor(reconstructed_img*255/
max(max(reconstructed_img)));
```

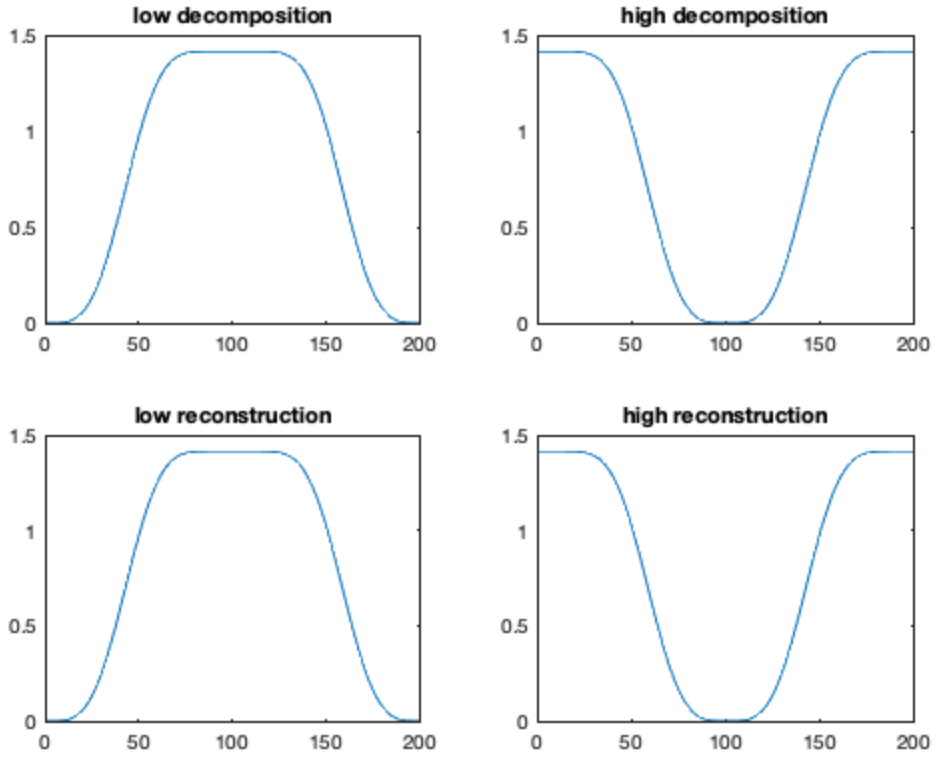
```
figure
imagesc(reconstructed_img)
axis('image')
axis('off')
colormap(gray(256))
title('reconstructed image')

difference_imgage = original_img - reconstructed_img;

figure
imagesc(difference_imgage)
axis('image')
axis('off')
colormap(gray(256))
title('input image and reconstruction image difference')


['---------SSIM output of Regenerated image (512*512->128*128-
>512*512) and original-------------------------']
['SSIM Output result for DB4 filter between
 Regenerated image and Original Image =
 ',num2str(100*ssim(reconstructed_img,original_img)),'%']


ans =

    '---------SSIM output of Regenerated image (512*512->128*128-
>512*512) and original-------------------------'


ans =

    'SSIM Output result for DB4 filter between Regenerated image and
 Original Image = 92.2495%'
```

reconstructed image



input image and reconstruction image difference

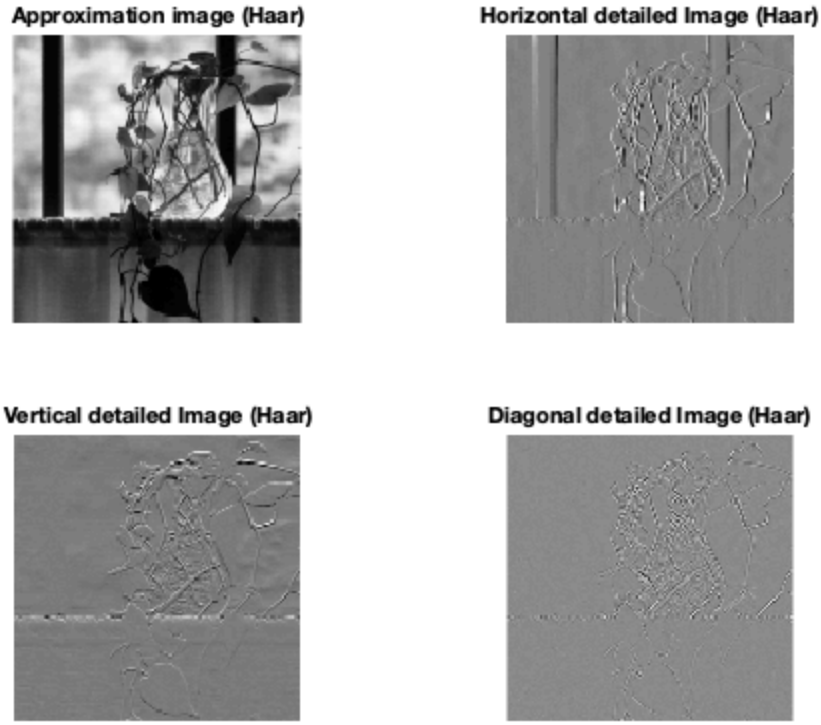
```
LFD_pad = zeros(200,1);
LFD_pad(1:8) = LFD;
ft_LFD = fft(LFD_pad);

HFD_pad = zeros(200,1);
HFD_pad(1:8) = HFD;
ft_HFD = fft(HFD_pad);

LFR_pad = zeros(200,1);
LFR_pad(1:8) = LFR;
ft_LFR = fft(LFR_pad);

HFR_pad = zeros(200,1);
HFR_pad(1:8) = HFR;
ft_HFR = fft(HFR_pad);

figure
subplot(2,2,1)
plot(fftshift(abs(ft_LFD)))
title('low decomposition')
subplot(2,2,2)
plot(fftshift(abs(ft_HFD)))
title('high decomposition')
subplot(2,2,3)
plot(fftshift(abs(ft_LFR)))
title('low reconstruction')
subplot(2,2,4)
plot(fftshift(abs(ft_HFR)))
title('high reconstruction')
```




```
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
% lets try haar experiment

[LFD,HFD,LFR,HFR] = wfilters('haar');

%apply h0 and h1 to columns of data
h0_row_output = zeros(row_len,col_len);
h1_row_output = zeros(row_len,col_len);
for c=1:col_len

    vec1 = original_img(:,c);  %get the columns sequentially

    % lets apply hi and low decomposition filters to column data
    h0_col_output(:,c) = conv(vec1,LFD,'same');
    h1_col_output(:,c) = conv(vec1,HFD,'same');

end;

%lets downsample the columns by a factor of 2 - these images are
 256x512
downsampled_h0 = h0_col_output(1:2:row_len,:);
downsampled_h1 = h1_col_output(1:2:row_len,:);

%lets repeat process for rows - note, result is 4 output arrays
```

```
%lets apply h0 and h1 to row of data for both h0 and h1 outputs

h0_h0_res = zeros(row_len/2,col_len);
h0_h1_res = zeros(row_len/2,col_len);
h1_h0_res = zeros(row_len/2,col_len);
h1_h1_res = zeros(row_len/2,col_len);

for r=1:row_len/2

    vec1 = downsampled_h0(r,:);  %get the columns sequentially
    h0_h0_res(r,:) = conv(vec1,LFD,'same');
    h0_h1_res(r,:) = conv(vec1,HFD,'same');

    vec1 = downsampled_h1(r,:);  %get the columns sequentially
    h1_h0_res(r,:) = conv(vec1,LFD,'same');
    h1_h1_res(r,:) = conv(vec1,HFD,'same');

end;

%downsample the columns by a factor of 2 - these images are 256x512 to
 get
%256*256 image
downsampled_h0_h0 = h0_h0_res(:,1:2:col_len);
downsampled_h0_h1 = h0_h1_res(:,1:2:col_len);
downsampled_h1_h0 = h1_h0_res(:,1:2:col_len);
downsampled_h1_h1 = h1_h1_res(:,1:2:col_len);


% lets use 256*256 image to decompose it to further 128*128 image

input_img_256_256=downsampled_h0_h0;


[row_len_1,col_len_1] = size(input_img_256_256);

for c=1:col_len_1

    vec1 = input_img_256_256(:,c);  %get the columns sequentially

    %apply hi and low decomposition filters to column data
    h0_col_output_1(:,c) = conv(vec1,LFD,'same');
    h1_col_output_1(:,c) = conv(vec1,HFD,'same');

end;

% lets downsample the columns by a factor of 2 - these images are
 128x256
ds_h0_1 = h0_col_output_1(1:2:row_len_1,:);
ds_h1_1 = h1_col_output_1(1:2:row_len_1,:);

%repeat process for rows - note, result is 4 output arrays

%apply h0 and h1 to row of data for both h0 and h1 outputs
```

```
h0_h0_res_1 = zeros(row_len_1/2,col_len_1);
h0_h1_res_1 = zeros(row_len_1/2,col_len_1);
h1_h0_res_1 = zeros(row_len_1/2,col_len_1);
h1_h1_res_1 = zeros(row_len_1/2,col_len_1);

for r=1:row_len_1/2

    vec1 = ds_h0_1(r,:);  %get the columns sequentially
    h0_h0_res_1(r,:) = conv(vec1,LFD,'same');
    h0_h1_res_1(r,:) = conv(vec1,HFD,'same');

    vec1 = ds_h1_1(r,:);  %get the columns sequentially
    h1_h0_res_1(r,:) = conv(vec1,LFD,'same');
    h1_h1_res_1(r,:) = conv(vec1,HFD,'same');

end;

%downsample the columns by a factor of 2 - these images are 128x256 to
 get
%128*128 image
downsampled_h0_h0_1 = h0_h0_res_1(:,1:2:col_len_1);
downsampled_h0_h1_1 = h0_h1_res_1(:,1:2:col_len_1);
downsampled_h1_h0_1 = h1_h0_res_1(:,1:2:col_len_1);
downsampled_h1_h1_1 = h1_h1_res_1(:,1:2:col_len_1);


%lets display Output
figure
subplot(2,2,1)
imagesc(downsampled_h0_h0_1)
axis('image')
axis('off')
colormap(gray(256))
title('Approximation image (Haar)')
subplot(2,2,2)
imagesc(downsampled_h0_h1_1)
axis('image')
axis('off')
colormap(gray(256))
title('Horizontal detailed Image (Haar)')
subplot(2,2,3)
imagesc(downsampled_h1_h0_1)
axis('image')
axis('off')
colormap(gray(256))
title('Vertical detailed Image (Haar)')
subplot(2,2,4)
imagesc(downsampled_h1_h1_1)
axis('image')
axis('off')
colormap(gray(256))
title('Diagonal detailed Image (Haar)')
```



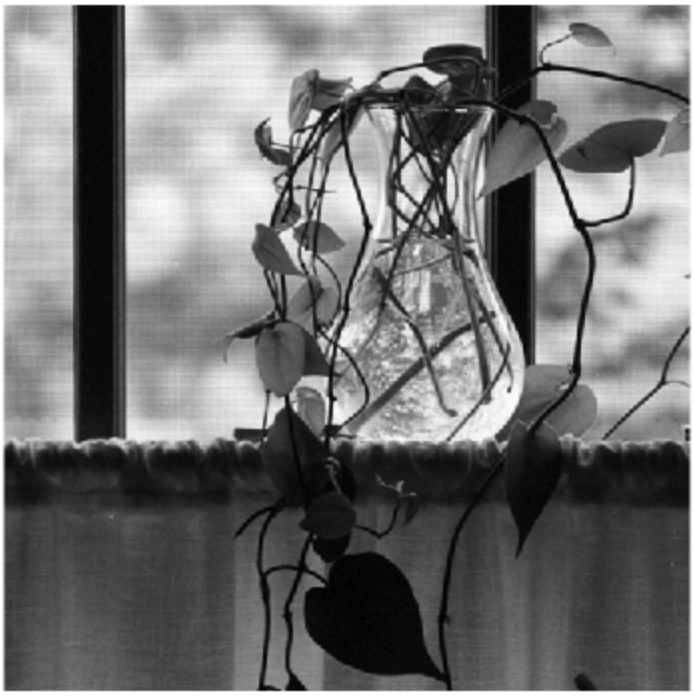

```
% lets start reconstruction of the image using 128*128 image from its
% decomposed components
%lets apply h0 and h1 to row of data for both h0 and h1 outputs

h0_h0_g0_input_1 = zeros(row_len_1/2,col_len_1);
h0_h1_g1_input_1 = zeros(row_len_1/2,col_len_1);
h1_h0_g0_input_1 = zeros(row_len_1/2,col_len_1);
h1_h1_g1_input_1 = zeros(row_len_1/2,col_len_1);

%lets upsample columns
for c=1:col_len_1/2

    h0_h0_g0_input_1(:,2*c) = downsampled_h0_h0_1(:,c);
    h0_h1_g1_input_1(:,2*c) = downsampled_h0_h1_1(:,c);
    h1_h0_g0_input_1(:,2*c) = downsampled_h1_h0_1(:,c);
    h1_h1_g1_input_1(:,2*c) = downsampled_h1_h1_1(:,c);

end;

for r=1:row_len_1/2

    vec1 = h0_h0_g0_input_1(r,:);  %get the columns sequentially
    h0_h0_g0_output_1(r,:) = conv(vec1,LFR,'same');

    vec1 = h0_h1_g1_input_1(r,:);  %get the columns sequentially
    h0_h1_g1_output_1(r,:) = conv(vec1,HFR,'same');
```

```
    vec1 = h1_h0_g0_input_1(r,:);  %get the columns sequentially
    h1_h0_g0_output_1(r,:) = conv(vec1,LFR,'same');

    vec1 = h1_h1_g1_input_1(r,:);  %get the columns sequentially
    h1_h1_g1_output_1(r,:) = conv(vec1,HFR,'same');

end;

%sum
h0_h0_g0_plus_h0_h1_g1_1 = h0_h0_g0_output_1 + h0_h1_g1_output_1;
h1_h0_g0_plus_h1_h1_g1_1 = h1_h0_g0_output_1 + h1_h1_g1_output_1;

% lets do upsampling and process rows
top_1 = zeros(row_len_1,col_len_1);
bottom_1 = zeros(row_len_1,col_len_1);

%lets upsample rows
for r=1:row_len_1/2

    top_1(2*r,:) = h0_h0_g0_plus_h0_h1_g1_1(r,:);
    bottom_1(2*r,:) = h1_h0_g0_plus_h1_h1_g1_1(r,:);

end;

output_top_1 = zeros(row_len_1,col_len_1);
output_bottom_1 = zeros(row_len_1,col_len_1);

for c=1:col_len_1

    vec1 = top_1(:,c);
    output_top_1(:,c) = conv(vec1,LFR,'same');

    vec1 = bottom_1(:,c);
    output_bottom_1(:,c) = conv(vec1,HFR,'same');

end;

recon_img_1 = output_top_1 + output_bottom_1;

%recon_img_1 is a 256*256 image reconsturction. lets use previous
%componenet to get the 512*512 image from 256*256 image

h0_h0_g0_ip = zeros(row_len/2,col_len);
h0_h1_g1_ip = zeros(row_len/2,col_len);
h1_h0_g0_ip = zeros(row_len/2,col_len);
h1_h1_g1_ip = zeros(row_len/2,col_len);

%lets upsample columns
for c=1:col_len/2

    h0_h0_g0_ip(:,2*c) = recon_img_1(:,c);
    h0_h1_g1_ip(:,2*c) = downsampled_h0_h1(:,c);
    h1_h0_g0_ip(:,2*c) = downsampled_h1_h0(:,c);
```

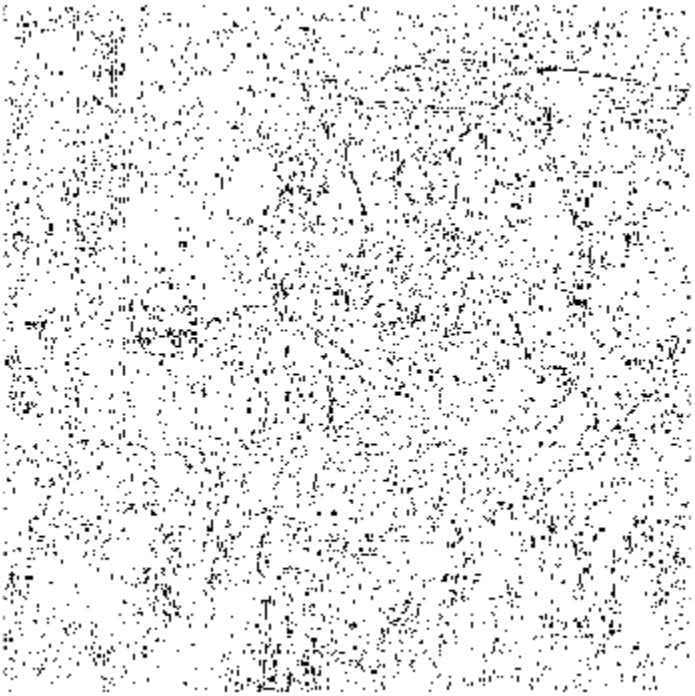
```
    h1_h1_g1_ip(:,2*c) = downsampled_h1_h1(:,c);

end;

for r=1:row_len/2

    vec1 = h0_h0_g0_ip(r,:);  %get the columns sequentially
    h0_h0_g0_output(r,:) = conv(vec1,LFR,'same');

    vec1 = h0_h1_g1_ip(r,:);  %get the columns sequentially
    h0_h1_g1_output(r,:) = conv(vec1,HFR,'same');

    vec1 = h1_h0_g0_ip(r,:);  %get the columns sequentially
    h1_h0_g0_output(r,:) = conv(vec1,LFR,'same');

    vec1 = h1_h1_g1_ip(r,:);  %get the columns sequentially
    h1_h1_g1_output(r,:) = conv(vec1,HFR,'same');

end;

%sum
h0_h0_g0_plus_h0_h1_g1 = h0_h0_g0_output + h0_h1_g1_output;
h1_h0_g0_plus_h1_h1_g1 = h1_h0_g0_output + h1_h1_g1_output;

%upsample and process rows
toplayer = zeros(row_len,col_len);
bottomlayer = zeros(row_len,col_len);

%upsample rows
for r=1:row_len/2

    toplayer(2*r,:) = h0_h0_g0_plus_h0_h1_g1(r,:);
    bottomlayer(2*r,:) = h1_h0_g0_plus_h1_h1_g1(r,:);

end;

output_top_layer = zeros(row_len,col_len);
output_bottom_layer = zeros(row_len,col_len);

for c=1:col_len

    vec1 = toplayer(:,c);
    output_top_layer(:,c) = conv(vec1,LFR,'same');

    vec1 = bottomlayer(:,c);
    output_bottom_layer(:,c) = conv(vec1,HFR,'same');

end;

reconstructed_img = output_top_layer + output_bottom_layer;

reconstructed_img = floor(reconstructed_img*255/
max(max(reconstructed_img)));
```

```
figure
imagesc(reconstructed_img)
axis('image')
axis('off')
colormap(gray(256))
title('reconstructed image (Haar)')

difference_imgage = original_img - reconstructed_img;

figure
imagesc(difference_imgage)
axis('image')
axis('off')
colormap(gray(256))
title('input image and reconstruction image difference (Haar)')

['---------SSIM output of Regenerated image (512*512->128*128-
>512*512) and original-------------------------']
['SSIM Output result for Haar filter between
 Regenerated image and Original Image =
 ',num2str(100*ssim(reconstructed_img,original_img)),'%']


ans =

    '---------SSIM output of Regenerated image (512*512->128*128-
>512*512) and original-------------------------'


ans =

    'SSIM Output result for Haar filter between Regenerated image and
 Original Image = 99.7954%'
```

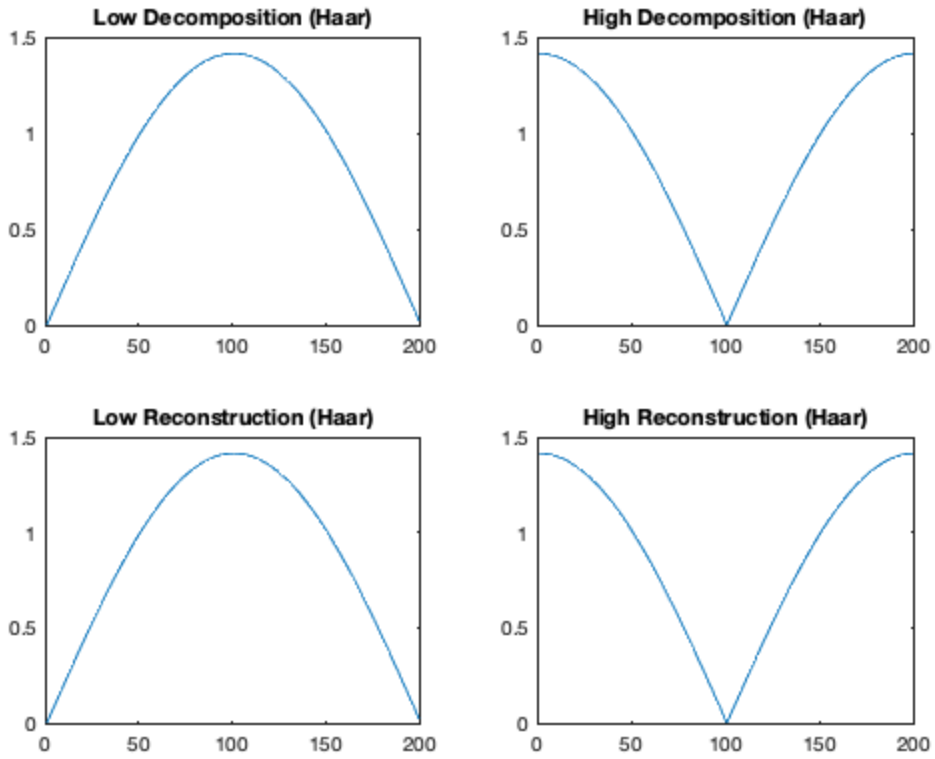
reconstructed image (Haar)



input image and reconstruction image difference (Haar)



```
LFD_pad = zeros(200,1);
LFD_pad(1:2) = LFD;
ft_LFD_1 = fft(LFD_pad);

HFD_pad = zeros(200,1);
HFD_pad(1:2) = HFD;
ft_HFD = fft(HFD_pad);

LFR_pad = zeros(200,1);
LFR_pad(1:2) = LFR;
ft_LFR = fft(LFR_pad);

HFR_pad = zeros(200,1);
HFR_pad(1:2) = HFR;
ft_HFR = fft(HFR_pad);

figure
subplot(2,2,1)
plot(fftshift(abs(ft_LFD_1)))
title('Low Decomposition (Haar)')
subplot(2,2,2)
plot(fftshift(abs(ft_HFD)))
title('High Decomposition (Haar)')
subplot(2,2,3)
plot(fftshift(abs(ft_LFR)))
title('Low Reconstruction (Haar)')
subplot(2,2,4)
plot(fftshift(abs(ft_HFR)))
title('High Reconstruction (Haar)')
```



Published with MATLAB® R2018b