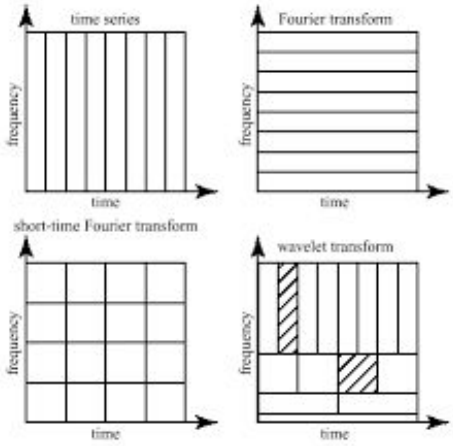
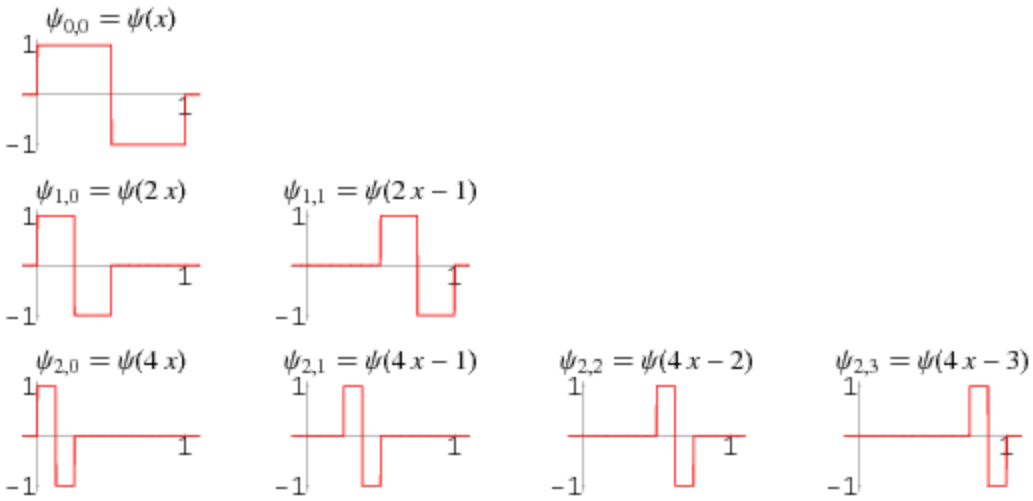
COL783 DIGITAL IMAGE ANALYSIS
ASSIGNMENT 3
Pyramids And Wavelets

Haar Wavelets and Denoising:-

Unlike Fourier transform whose basis function is sinusoidal waves, Wavelets transforms are based on small waves of limited duration. In Fourier transform we have only frequency resolution information but we don't have any time resolution, therefore, that is the biggest disadvantage of Fourier transform. Since two completely different signals can have same frequency patterns in frequency domain but we do not know when do these frequency appears in signal using Fourier transform. Wavelets overcome this problem by time and frequency information of a signal at the same time. The idea behind the time-frequency joint representations is to cut the signal of interest into several parts and then analyze the parts separately.





Haar Wavelets Representation

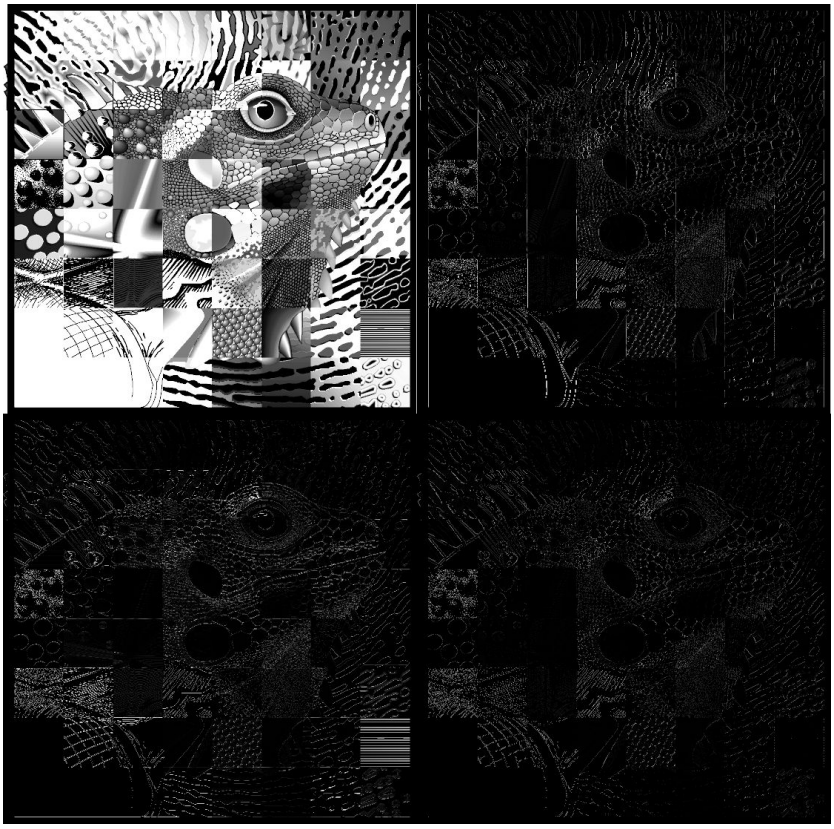
Haar Transform

Basically wavelet transform provides details and approximation coefficients of a signal, for an image, approximation coefficients are all the smooth regions which consider all of the low frequency details and details coefficients are high frequency components(such as edges for image case)

We represent an image in following way after discrete haar transform:

LL	LH
HL	HH

LL- approximation coefficient LH - horizontal details coefficients
HL - vertical details coefficients HH - diagonal details coefficients



And iteratively we can get details coefficients and approximation at this scale(top left image) which can be represented by this

Haar Transform at all scales

Inverse Haar Transform

Original Image

Reconstructed Image(ihaar)

Denoising

Gaussian noise of variance - 0.01, mean =0

DENOISED IMAGE after manual thresholding at for detail coefficients at - **0.3(Hard thresholding)**

Psnr - 19.72

Psnr - 19.14 (for same threshold value and soft Thresholding)

Image Compression

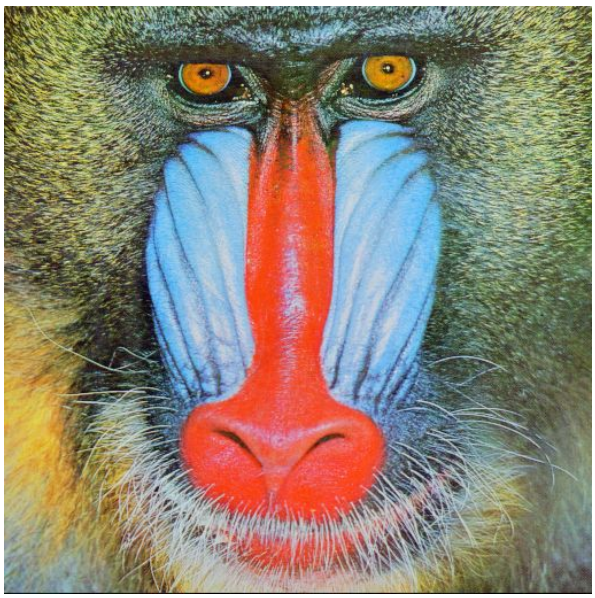
We implemented simple image compression programme, our implementation uses only run length encoding after thresholding of detail coefficients. For thresholding of detail coefficients we take an input K in range of 0-100 and retain only top K% detail coefficients which ensure the spare detail coefficients

Our run length encoding schemes - we iterate over each coefficient of 2D matrix if it is not zero we store the value in string and if there are multiple zeros in between 2 non zero value, we count the number of zeros and store in string
"**coefficient_value number_of_zeros_in_between**" and " **;** " for end of line

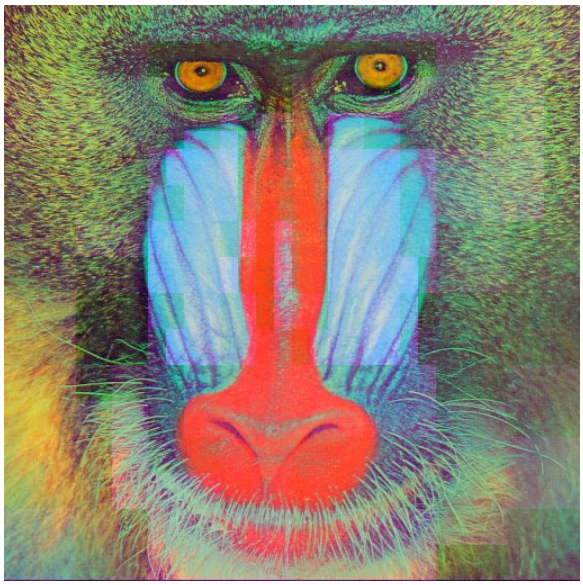
For example

0.1	0.2	0	0	0	-0.01	0	0	0	0

For this case our string will be following and save this string in .txt file
"**0.1 0 0.2 3 -0.01 4 ;**"



Original Image
Size - 622KB



Uncompressed Image after compressing
size of compressed file - 1.04KB
K = 50%

Size - 370 KB (original image)

Inverse haar after thresholding K - 55% Size of compressed file -1.02KB

Image Pyramids and Image Blending:-

We constructed the Gaussian Pyramid by blurring and downsampling the image using the gaussian kernel as described in Burt and Adelson (1983). Blurring was performed by convolution of the image with a gaussian filter. Reduce and expand operations were performed as given in the paper to create a pyramid having four levels. Once the gaussian pyramid has been constructed we performed the following operation to construct the Laplacian Pyramid for each level l:

$$LP[l] = GP[l] - expanded(GP[l-1])$$

Gaussian Pyramid

256x256 **128x128** **64x64** **32x32**

Laplacian Pyramid

256x256 **128x128** **64x64** **32x32**

Original Image

Reconstructed Image

Image Blending

1.) Blending with Overlapping Regions

This was by constructing the laplacian pyramids of original images and then taking the left region of laplacian pyramid of image1 and right region of laplacian pyramid of image2 for every level such that

LS(l) = L1(l)[0:col/2] + L2[col/2:]

Original Images

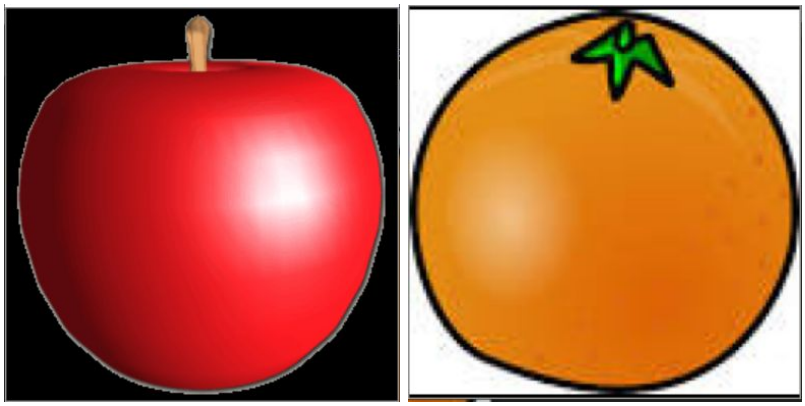
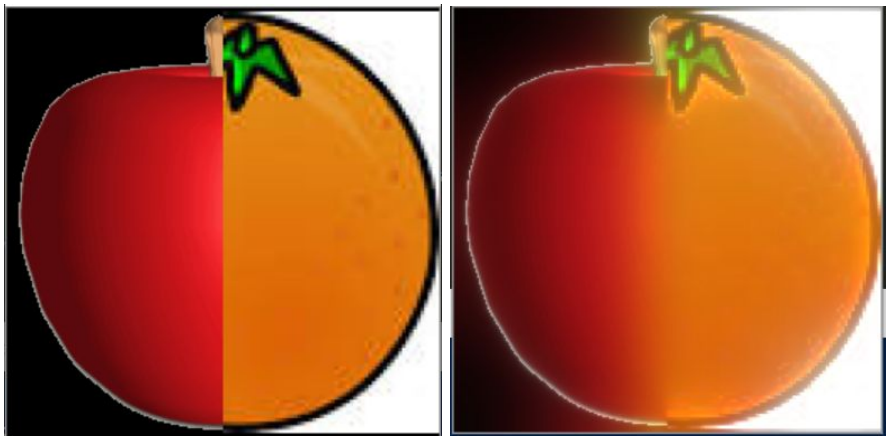


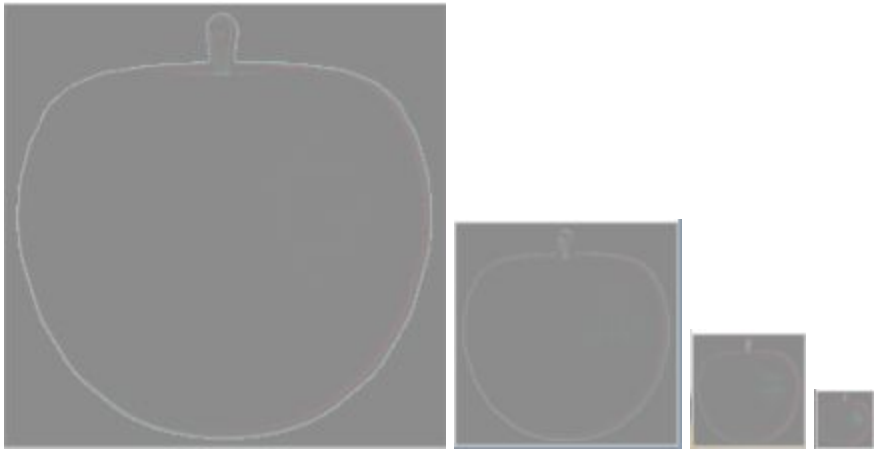
Image1: 256x 256 **Image2: 256x256**

Results

Left image is without blending and right image is constructed after blending



Laplacian Pyramids



256x256 **128x128** **64x64** **32x32**

256x256 **128x128** **64x64** **32x32**

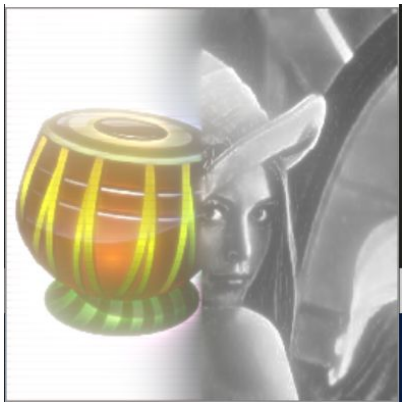
Original Images


Image1: 256x256 **Image2: 256x256**

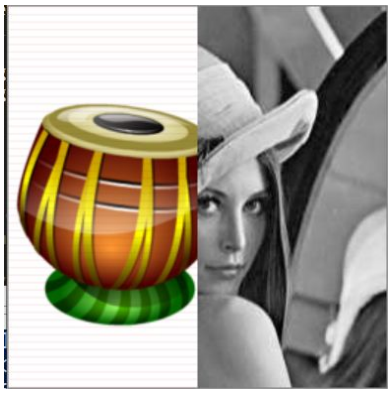
Laplacian of Image2




256x256 **128x128** **64x64** **32x32**



Result after using pyramids



Result without using pyramids

2.)Blending with Arbitrary Regions

Original Images



Image 1 **Image 2** **Mask**

Step 1: Laplacian pyramids of Image1 and Image2 were created such that laplacian pyramid of Image1 was LP1 and that of Image2 was LP2

Step 2: Gaussian pyramid of the mask was created (after normalizing the intensity of the binary mask) such that it was GP

Step 3: A third laplacian pyramid LP3 was constructed such that for each level of LP3

$$LP3 = (GP*LP1) + (1-GP)*LP2$$

Laplacian pyramid of Image 2:

Similarly Laplacian of Image 1 was calculated

Reconstructed images

Resulting Image