DIGITAL IMAGE PROCESSING ASSIGNMENT

Name -> Arpit Sharma

Course -> B.Sc (Hons.) Computer Science

Roll No. -> 20201406

Exam Roll No. -> 20020570008

Q:1) Implement the JPEG compression algorithm. JPEG uses the DCT Transform. Create three versions of the algorithm:

1) Using the standard DCT Transform

2) Using Fourier Transform

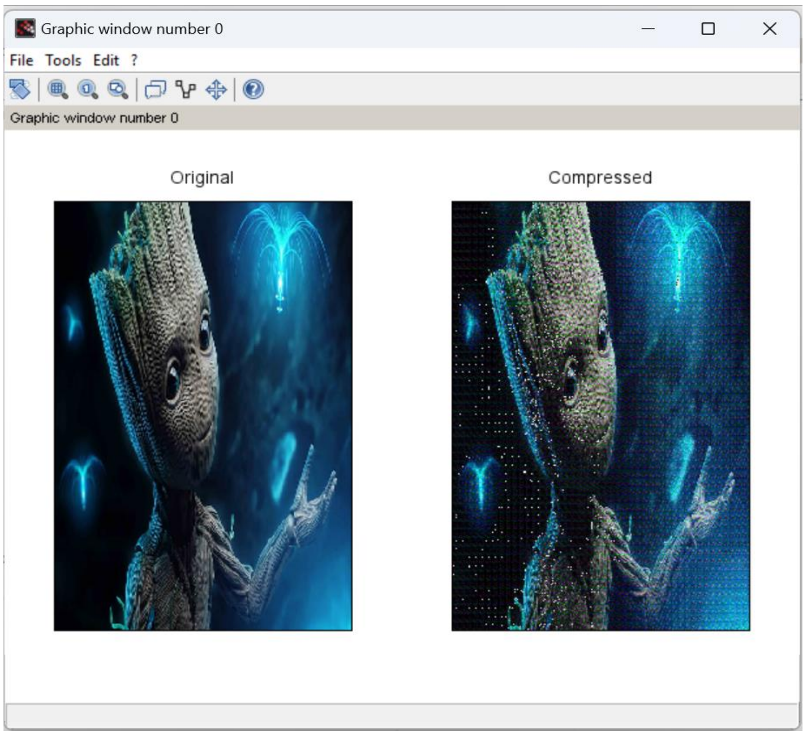
3) Using Wavelet Transform.

Compare the performance of the three versions after application of the compression scheme on an image. The comparison will be on the basis of the standard compression performance indices used. (Paste your code and output in the file to be submitted)

Solution:
1(a) Standard DCT Algorithm :-
The discrete cosine transform (DCT) is a technique for converting a signal into elementary
frequency components. Like other transforms, the Discrete Cosine Transform (DCT)
attempts
to de correlate the image data. After de correlation each transform coefficient can be
encoded
independently without losing compression efficiency.
DCT Algorithm:
•The following is a general overview of the JPEG process.
•The image is broken into 8x8 blocks of pixels.
•Working from left to right, top to bottom, the DCT is applied to each block.
•Each block is compressed through quantization.
•The array of compressed blocks that constitute the image is stored in a drastically
reduced amount of space.

•When desired, the image is reconstructed through decompression, a process that uses the inverse Discrete Cosine Transform (IDCT)

```
I = imread('C:\Users\dives\Downloads\groot.jpg');
Y_d = rgb2ycbcr( I );
// Downsample:
Y_d(:,:,2) = 2*round(Y_d(:,:,2)/2);
Y_d(:,:,3) = 2*round(Y_d(:,:,3)/2);

Q = [16 11 10 16 24 40 51 61;
   12 12 14 19 26 28 60 55;
   14 13 16 24 40 57 69 56;
   14 17 22 29 51 87 80 62;
   18 22 37 56 68 109 103 77;
   24 35 55 64 81 104 113 92;
   49 64 78 87 103 121 120 101;
   72 92 95 98 112 100 103 99];

// DCT compress:
B = zeros(size(Y_d));
for channel = 1:3
  for j = 1:8:size(Y_d,1)-7
     for k = 1:8:size(Y_d,2)
        II = double(Y_d(j:j+7,k:k+7,channel));
        freq = dct(dct(II).').';
        freq = Q.*round(freq./Q);
        B(j:j+7,k:k+7,channel) = idct(idct(freq).').';
     end
  end
end
subplot(1,2,1)
imshow(ycbcr2rgb(Y_d))
title('Original')
subplot(1,2,2)
imshow(ycbcr2rgb(uint8(B)));
title('Compressed')
```

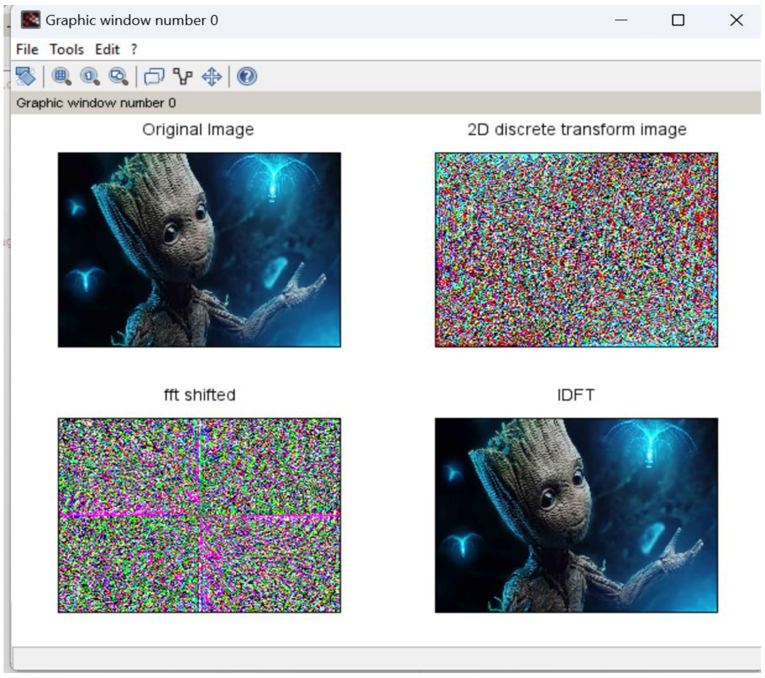


1(b)

The Discrete Fourier transform (DFT) is applied to each M x N block independently to represent the image in the frequency domain yielding the real and imaginary components. The Matrix Minimization algorithm is applied to each component and zeros are removed. The resulting vectors are subjected to Arithmetic coding and represent the compressed data.

A uniform quantization is then applied to both parts, which involves dividing each element by a factor called quantization factor Q followed by rounding the outcomes which results in an increase of high frequency coefficients probability thus reducing the number of bits needed to represent such coefficients. The result of this operation is that the compression ratio increases.

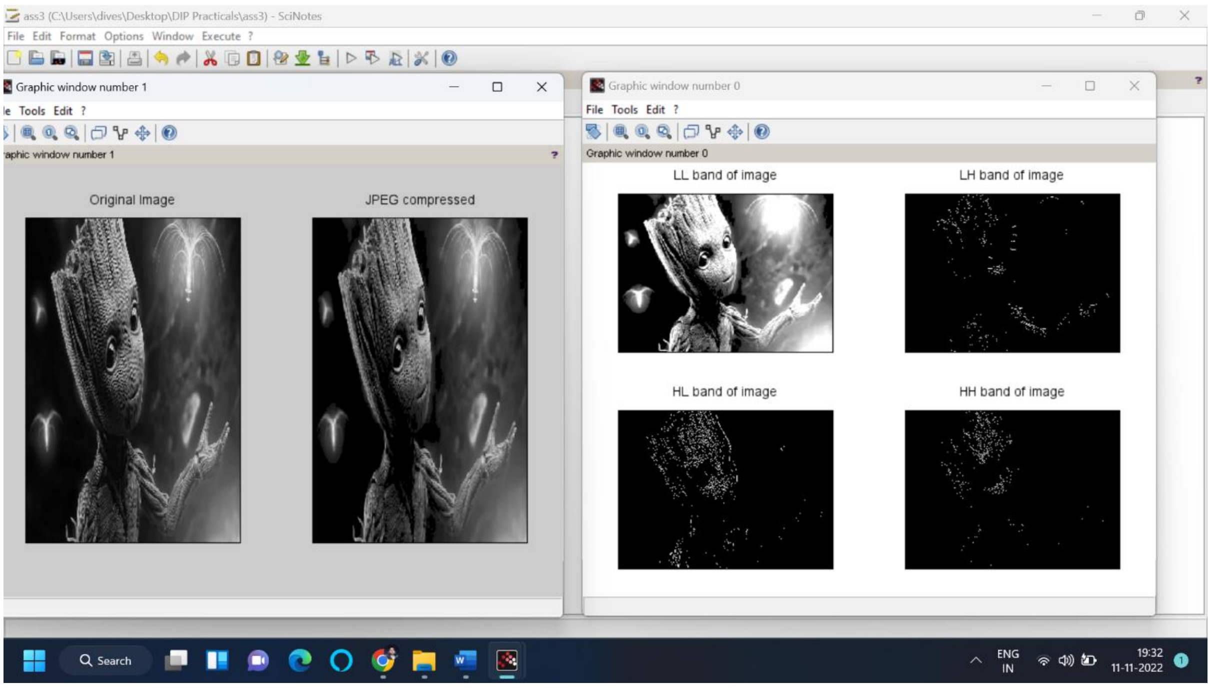
```
a = imread('C:\Users\dives\Downloads\groot.jpg') ;
[m ,n ]= size (a);
subplot(221);
imshow(a);
title("Original Image" ) ;
A = fft(double(a));
subplot(222);
imshow(A);
title('2D discrete transform image');
B = fftshift(A);
subplot(223);
imshow(B);
title('fft shifted' ) ;
a_inv = real(ifft(A));
subplot(224);
imshow( uint8(a_inv ));
title('IDFT') ;
```



1(c)

The wavelet analysis method is a time-frequency analysis method which selects the appropriate frequency band adaptively based on the characteristics of the signal. Then the frequency band matches the spectrum which improves the time-frequency resolution.

```
rgb = rgb2gray(imread('C:\Users\dives\Downloads\groot.jpg'));
[LL,LH,HL,HH] = dwt2(double(rgb), "haar");

LL(LL < std2(rgb)) = 0;
LH(LH < std2(rgb)) = 0;
HL(HL < std2(rgb)) = 0;
HH(HH < std2(rgb)) = 0;

subplot(2,2,1);imshow(LL/255);title('LL band of image');
subplot(2,2,2);imshow(LH);title('LH band of image');
subplot(2,2,3);imshow(HL);title('HL band of image');
subplot(2,2,4);imshow(HH);title('HH band of image');
figure;
subplot(1,2,1);
imshow(rgb);
title('Original Image')
subplot(1,2,2)
imshow(uint8(idwt2(LL,LH,HL,HH,'haar')));
title('JPEG compressed');
```



Q2) Refer to the paper **Sarma, Rituparna, and Yogesh Kumar Gupta. "A comparative study of new and existing segmentation techniques."** ***IOP Conference Series: Materials Science and Engineering*****. Vol. 1022. No. 1. IOP Publishing, 2021. It presents a number of image segmentation techniques; select any of these and describe the working of the algorithm. Implement and apply the algorithm on an image.**

Solution:

Automatic global thresholding algorithms usually have following steps.

Process the input image
Obtain image histogram (distribution of pixels)
Compute the threshold value T
Replace image pixels into white in those regions, where saturation is greater than T and into the black in the opposite cases.

Algorithm for otsu thresholding :

1. calculate the histogram and intensity level probabilities
2. initialize $w_i(0), \mu_i(0)$
3. iterate over possible thresholds: $t = 0, \ldots, max\_intensity$
   - update the values of w_i, μ_i, where w_i is a probability and μ_i is a mean of class i
   - calculate the between-class variance value $\sigma_b^2(t)$
4. the final threshold is the maximum $\sigma_b^2(t)$ value

$w_i(0), \mu_i(0)$ = Probability and Mean of occuring pixel
$\sigma_b^2(t)$ = Interclass Variance

Otsu's method performs well when the histogram has a bimodal distribution with a deep and sharp valley between the two peaks.

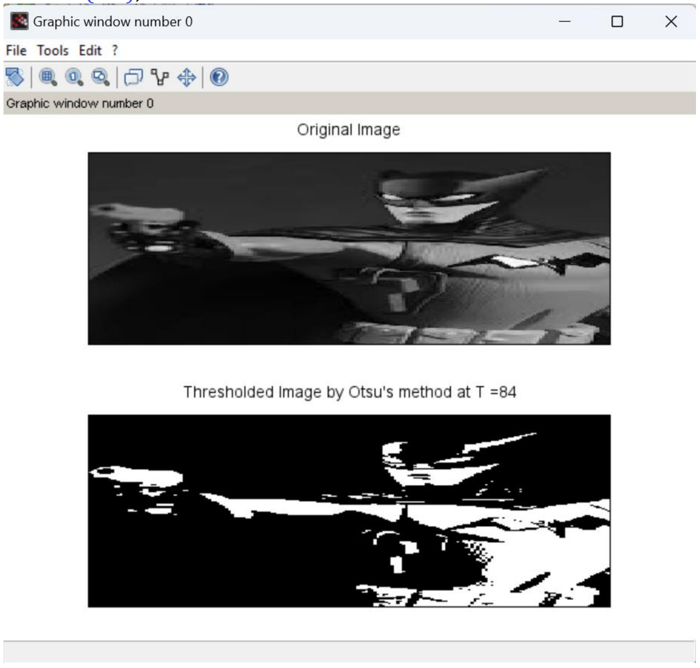
Like all other global thresholding methods, Otsu's method performs badly in case of heavy noise, small objects size, inhomogeneous lighting and larger intra-class than inter-class variance. In those cases, local adaptations of the Otsu method have been developed.

Moreover, the mathematical grounding of Otsu's method models the histogram of the image as a mixture of two Normal distributions with equal variance and equal size. Otsu's thresholding may however yield satisfying results even when these assumptions are not met, in the same way statistical tests (to which Otsu's method is heavily connected) can perform correctly even when the working assumptions are not fully satisfied. However, several variations of Otsu's methods have been proposed to account for more severe deviations from these assumptions, such as the Kittler-Illingworth method.

Code:

```
function level=otsu_threshold(I)
if (size(I,3) == 3) then
   I = rgb2gray(I);
end

// Calculation of the normalized histogram
n = 256;
h = imhist(I(:), n);
h = h/(length(I(:)) + 1);

// Calculation of the cumulated histogram and the mean values
w = cumsum(h);
mu = zeros(n, 1);
mu(1) = h(1);
for i=2:n
   mu(i) = mu(i-1) + i*h(i);
end

// Initialisation of the values used for the threshold calculation
level = find (h > 0, 1);
w0 = w(level);
w1 = 1-w0;
mu0 = mu(level)/w0;
mu1 = (mu($)-mu(level))/w1;
maxval = w0*w1*(mu1-mu0)*(mu1-mu0);

// For each step of the histogram
// calculation of the threshold and storing of the maximum
for i = find (h > 0)
   w0 = w(i);
   w1 = 1-w0;
   mu0 = mu(i)/w0;
   mu1 = (mu($)-mu(i))/w1;
   s = w0*w1*(mu1-mu0)*(mu1-mu0);
   if (s > maxval)
      maxval = s;
      level = i;
   end
end

// Normalisation of the threshold
level = level./n;
endfunction
img = imread('C:\Users\dives\Downloads/btmn.jfif');
img = rgb2gray(img);
subplot(2,1,1), title('Original Image'), imshow(img);
```

```
th = otsu_threshold(img);
bw = im2bw(img, th);
subplot(2,1,2);
title('Thresholded Image by Otsu''s method at T =' + string(th*256));
imshow(bw);
```



Original Image

Thresholded Image by Otsu's method at T =84

Q3) Refer to any recent research publication on low pass/high pass filtering on images. Describe the algorithm used. Present an analysis of the same and enumerate its strength and weaknesses. Extra points for implementing the algorithm used. Papers can be found at Google Scholar.

Solution:

This paper proposes a new technique based on nonlinear Minmax Detector Based (MDB) filter for image restoration.The aim of image enhancement is to reconstruct the true image from the corrupted image. The process of image acquisition frequently leads to degradation and the quality of the digitized image becomes inferior to the original image.

Proposed Algorithms

Algorithm: MDB Filter

Step 1. Take corrupted image (X).

Step 2. Take a 3 x 3 window (W).

Let the center pixel be the test pixel.

Step 3. Shift the window row wise then column

wise to cover the entire pixel in the image

and repeat Step4 and Step5.

Step 4. If (test pixel < min (rest of the pixel in

W) OR

(test pixel > max (rest of the pixel in W)

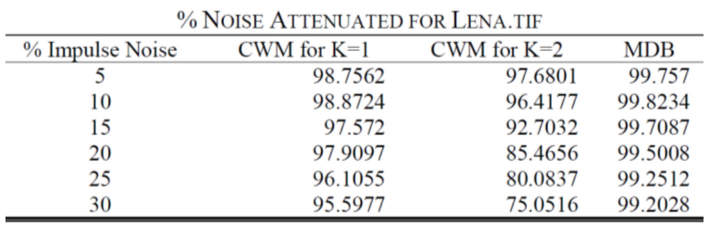
then the test pixel is corrupted.

Step 5. If the test pixel is corrupted apply median

filter to the test filter in the window W.

Step 6. Stop.

The quantitative results has been given in table for the standard LENA image, for different percentage of noise, starting from 5% to 30% with a step 5%, and the comparative analysis has been presented in figure for LENA image showing the performance of CWM(Central Weighted Median) filters for gain =1 and 2 respectively from 5% to 30% along with Salt and Pepper noise. Along with the figures and tables some graphs, shown in figure,has also been given, for all the quantitative measures, for LENA image to have a quick insight into the comparative performance of the existing filters along with the proposed one i.e. MDB filter.



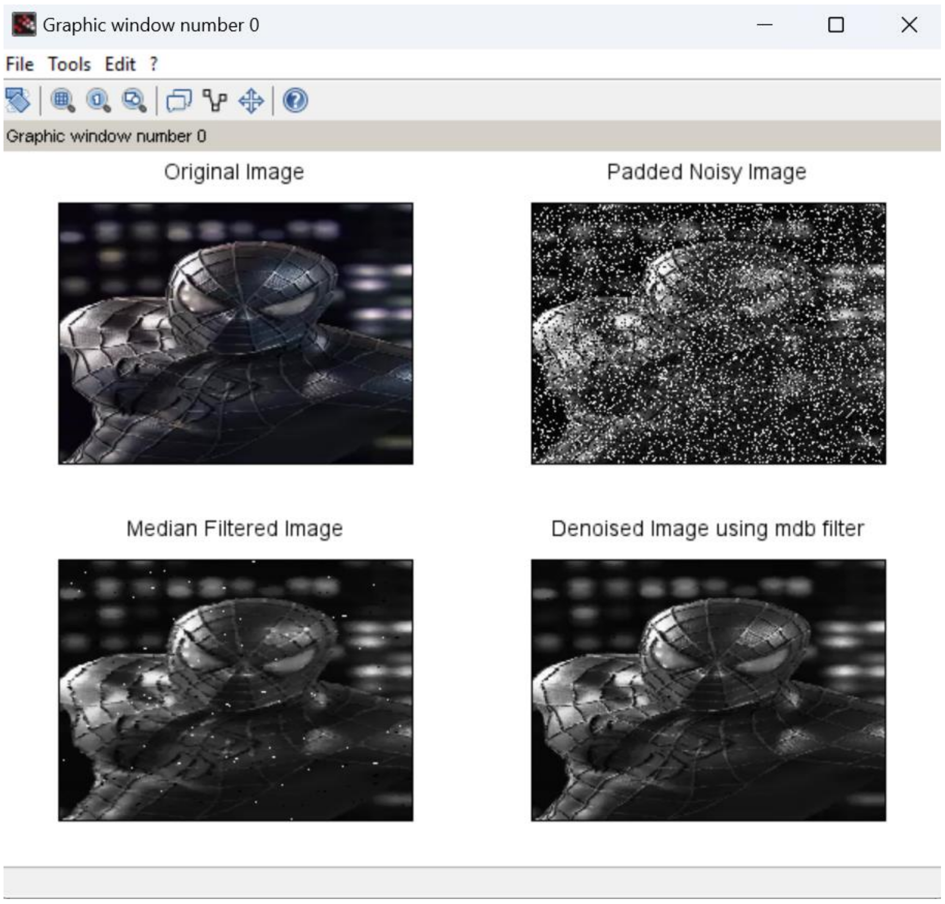
% NOISE ATTENUATED FOR LENA.TIF

% Impulse Noise	CWM for K=1	CWM for K=2	MDB
5	98.7562	97.6801	99.757
10	98.8724	96.4177	99.8234
15	97.572	92.7032	99.7087
20	97.9097	85.4656	99.5008
25	96.1055	80.0837	99.2512
30	95.5977	75.0516	99.2028

Code:

```
// mdb (MinMax Based Detector) filter

// Original Image
grayImg = imread('C:\Users\dives\Downloads/spiman.webp');
subplot(221), title("Original Image"), imshow(grayImg);
grayImg=rgb2gray(grayImg);
// Noised Image
d_im = imnoise(grayImg, 'salt & pepper', 0.25);
subplot(222), title("Padded Noisy Image"), imshow(d_im);

// algorithm
[r c] = size(d_im);
img1 = zeros(r+2, c+2, 'uint8');
img1(2:r+1, 2:c+1) = d_im(:,:);

// border padded image
img1(1, 1) = d_im(1, 1);
img1(r+2, 1) = d_im(r, 1);
img1(1, c+2) = d_im(1, c);
```

```
img1(r+2, c+2) = d_im(r, c);

img1(2:r+1, 1) = d_im(:,1);
img1(2:r+1, c+2) = d_im(:,c);
img1(1, 2:c+1) = d_im(1,:);
img1(r+2, 2:c+1) = d_im(r,:);

subplot(223), title("Median Filtered Image"), imshow(immedian(d_im,3));
for i = 2:r+1
   for j = 2:c+1
      if img1(i,j) == 0 | img1(i,j) == 255 then
         img1(i,j) = gsort(img1(i-1:i+1, j-1:j+1))(5);
      end
   end
end
subplot(224), title("Denoised Image using mdb filter"), imshow(img1(2:r+1, 2:c+1));
```



THANK YOU