clc

%Clear the Workspace

clear all

%Display the Workspace

%workspace

%Set unwanted minor warnings off

% warning off

% my image.jpg

% Start of Program

% Enter the flowchart of the proposed method

% disp('Browse the Flowchart of the Proposed method')

% [flowchart , pathname]= uigetfile('\*.bmp;\*.BMP;\*.tif;\*.TIF;\*.jpg;\*.gif','Browse the Proposed method Flowchart');

% flow\_chart=imread(char(flowchart));

% imshow(flow\_chart);

% title('Flowchart of the Proposed Method');

% disp('Press Any Key To Continue')

% pause

Image=imread('my image.jpg');

%Displaying Input Image

figure;imshow(Image);

%displaying the pixel values of the IMAGE

impixelregion;

title('Input Image');

%msgbox('INPUT IMAGE IS DISPLAYED')

% Checking Whether the INPUT Image is RGB or not

%If RGB then Convert into Gray Scale

[m,n,z]=size(Image); % Size of Image

% m is No. of Row Pixels

% n is No. of Column Pixel

% z is the Color Space Value

if z==3

disp('Program is Converting RGB to Gray...........')

Gray\_Image=rgb2gray(Image);

disp('RGB Image Has been Converted to Gray Scale')

else

Gray\_Image=Image;

end

figure,imshow(Gray\_Image);impixelregion;

title('Gray Scale Image');

%msgbox('GRAY SCALE IMAGE IS DISPLAYED')

% Convert Image to type Double

disp(' ')

disp('Program is Converting Image to type Double...........')

Double\_Converted\_Image=im2double(Gray\_Image);

disp('Gray Image Has been Converted to type Double')

figure,imshow(Double\_Converted\_Image);

title('Double Converted Image');

%msgbox('DOUBLE CONVERTED IMAGE IS DISPLAYED')

impixelregion;

disp('Press Any Key to Start Decomposition')

pause

X=Double\_Converted\_Image;

% Decomposition of Image

% The db1 filter is used.

wname = 'db1';

% Compute a 3-level decomposition of the image using the db filters.

[wc,s] = wavedec2(X,3,wname);

% Extract the level 1 coefficients.

a1 = appcoef2(wc,s,wname,1);

h1 = detcoef2('h',wc,s,1);

v1 = detcoef2('v',wc,s,1);

d1 = detcoef2('d',wc,s,1);

% Extract the level 2 coefficients.

a2 = appcoef2(wc,s,wname,2);

h2 = detcoef2('h',wc,s,2);

v2 = detcoef2('v',wc,s,2);

d2 = detcoef2('d',wc,s,2);

% Extract the level 3 coefficients.

a3 = appcoef2(wc,s,wname,3);

h3 = detcoef2('h',wc,s,3);

v3 = detcoef2('v',wc,s,3);

d3 = detcoef2('d',wc,s,3);

% Display the decomposition up to level 1 only.

sz = size(X);

cod\_a1 = wcodemat(a1); cod\_a1 = wkeep(cod\_a1, sz/2);

cod\_h1 = wcodemat(h1); cod\_h1 = wkeep(cod\_h1, sz/2);

cod\_v1 = wcodemat(v1); cod\_v1 = wkeep(cod\_v1, sz/2);

cod\_d1 = wcodemat(d1); cod\_d1 = wkeep(cod\_d1, sz/2);

disp('Image after 1st level of Decomposition')

figure;image([cod\_a1,cod\_h1;cod\_v1,cod\_d1]);

axis image; set(gca,'XTick',[],'YTick',[]); title('Single stage decomposition')

% pause

% Display the entire decomposition upto level 2.

cod\_a2 = wcodemat(a2); cod\_a2 = wkeep(cod\_a2, sz/4);

cod\_h2 = wcodemat(h2); cod\_h2 = wkeep(cod\_h2, sz/4);

cod\_v2 = wcodemat(v2); cod\_v2 = wkeep(cod\_v2, sz/4);

cod\_d2 = wcodemat(d2); cod\_d2 = wkeep(cod\_d2, sz/4);

disp('Image after 2nd level of Decomposition')

% figure;image([[cod\_a2,cod\_h2;cod\_v2,cod\_d2],cod\_h1;cod\_v1,cod\_d1]);

figure;image([cod\_a2,cod\_h2;cod\_v2,cod\_d2]);

axis image; set(gca,'XTick',[],'YTick',[]); title('Two stage decomposition')

% pause

% Display the entire decomposition upto level 3.

cod\_a3 = wcodemat(a3); cod\_a3 = wkeep(cod\_a3, sz/8);

cod\_h3 = wcodemat(h3); cod\_h3 = wkeep(cod\_h3, sz/8);

cod\_v3 = wcodemat(v3); cod\_v3 = wkeep(cod\_v3, sz/8);

cod\_d3 = wcodemat(d3); cod\_d3 = wkeep(cod\_d3, sz/8);

disp('Image after 3rd level of Decomposition')

% figure;image([[[cod\_a3,cod\_h3;cod\_v3,cod\_d3],cod\_h2;cod\_v2,cod\_d2],cod\_h1;cod\_v1,cod\_d1]);

figure;image([cod\_a3,cod\_h3;cod\_v3,cod\_d3]);

axis image; set(gca,'XTick',[],'YTick',[]); title('Three stage decomposition')

% pause

[Lo\_D,Hi\_D,Lo\_R,Hi\_R]=wfilters('db1');

% Computes 2d Wavelet Transformation

a1=Double\_Converted\_Image(:,:);

a2=(log(10))\*(1/3);

a3=a2\*(Lo\_D.^2);

a4=a2\*(Hi\_D.^2);

a5=a2\*(Lo\_R.^2);

a6=a2\*(Hi\_R.^2);

% Calculation of Log-Energy of Each Subband

LH=a4;

HL=a5;

HH=a6;

% W is Weight of Energy at HH Subband

W=0.8;

% Calculation of Log Energy at each decomposition

% TLE are the pre-level log-energy values

TLE = ((1-W)\*((LH+HL)/2))+(W\*HH); % Equation '1'

% Calculation of the Scalar Sharpness Index (SSI)

SSIt=0.0;

for n=1:3

SSIn=SSIt+((2^(3-n))-TLE); % Equation '2'

SSIt=SSIn;

end

SSI=SSIt;

% disp('Press Any Key to Start SSI Filtering')

% pause

% Application of SSI Filter on Image

disp(' ')

% disp('Program is Filtering Image by Applying SSI Filter...........')

After\_SSI=imfilter(Double\_Converted\_Image,SSI);

% disp('Image has been Filtered Out after Applying SSI Filter')

% figure,imshow(After\_SSI);

% title('Image after SSI Filtering to Double Conveted Image');

%msgbox('After SSI Filtering')

% impixelregion;

rgbImage=Image;

% Test code if you want to try it with a gray scale image.

% Uncomment line below if you want to see how it works with a gray scale image.

% rgbImage = rgb2gray(rgbImage);

% Display image full screen.

% Enlarge figure to full screen.

% Get the dimensions of the image. numberOfColorBands should be = 3.

[rows columns numberOfColorBands] = size(rgbImage);

% Block based division

blockSizeR = input('Enter the no.of Rows of the block = '); % Rows in block.

blockSizeC = input('Enter the no.of Columns of the block = '); % Columns in block.

% Figure out the size of each block in rows.

% Most will be blockSizeR but there may be a remainder amount of less than that.

wholeBlockRows = floor(rows / blockSizeR);

blockVectorR = [blockSizeR \* ones(1, wholeBlockRows), rem(rows, blockSizeR)];

% Figure out the size of each block in columns.

wholeBlockCols = floor(columns / blockSizeC);

blockVectorC = [blockSizeC \* ones(1, wholeBlockCols), rem(columns, blockSizeC)];

% Create the cell array, ca.

% Each cell (except for the remainder cells at the end of the image)

% in the array contains a blockSizeR by blockSizeC by 3 color array.

% This line is where the image is actually divided up into blocks.

if numberOfColorBands > 1

% It's a color image.

ca = mat2cell(rgbImage, blockVectorR, blockVectorC, numberOfColorBands);

else

ca = mat2cell(rgbImage, blockVectorR, blockVectorC);

end

% Now display all the blocks.

plotIndex = 1;

numPlotsR = size(ca, 1);

numPlotsC = size(ca, 2);

figure;

for r = 1 : numPlotsR

for c = 1 : numPlotsC

fprintf('plotindex = %d, c=%d, r=%d\n', plotIndex, c, r);

% Specify the location for display of the image.

subplot(numPlotsR, numPlotsC, plotIndex);

% Extract the numerical array out of the cell

rgbBlock = ca{r,c};

imshow(rgbBlock); % Can call imshow(ca{r,c})

[rowsB columnsB numberOfColorBandsB] = size(rgbBlock);

% Make the caption the block number.

caption = sprintf('%d of %d\n%d rows by %d columns'); ...

title(caption);

drawnow;

% Increment the subplot to the next location.

plotIndex = plotIndex + 1;

end

end

% Display the original image in the upper left.

subplot(4, 6, 1);

imshow(rgbImage);

title('Original Image');

% Sharpness Estimator by applying Local Block Based SSI

T=plotIndex;

BSSIt=(SSI.\*SSI);

disp('Press Any Key to Start BSSI Filtering')

% pause

% Block Based SSI Sharpness Index

tic

for i=1:T

BSSI=(sqrt((1/T)\*BSSIt)); % Equation '3'

end

toc

% Application of Block Based SSI Filter on Image

After\_BSSI=imfilter(Double\_Converted\_Image,BSSI);

disp('Image has been Filtered Out after Applying Block Based SSI Filter(BSSI)')

figure, imshow(After\_BSSI);

title('Image after Block Based SSI (BSSI) Filtering to Double Conveted Image');

impixelregion;

%msgbox('After BSSI Filtering')

disp(' ')

% End of Program