clc

clear all;

close all;

% [I1]=imgetfile;

%% i1=im2double(rgb2gray(imread(I1)));

i1=imread('lena\_color.tiff');

i1=i1(1+5:494+5,1+5:494+5,:);

i1=im2double(rgb2gray(i1));

%% DWT transfoem at level 1

[cA1,cH1,cV1,cD1] = dwt2(i1,'sym4');

figure, imshow(i1),title('Original Image (I) ');

figure,

subplot(2,2,1);imshow(cA1);title('CA1');

subplot(2,2,2);imshow(cH1);title('CH1');

subplot(2,2,3);imshow(cV1);title('CV1');

subplot(2,2,4);imshow(cD1);title('CD1');

cA, cV, cH, cD = UNIV\_Thres(cA, cV, cH, cD );

%% First stage compressoion using entropy

% [x,y]=size(cA1);

% disp(x);

% disp(y);

% for i=1:x-4

% for j=1: y-4

% submatrix = cA1(i:i+4, j:j+4);

% entpy=entropyfilt(submatrix);

% if (entpy<std2(submatrix)\*sqrt(2\*log(sum(sum(submatrix>0)))))

% md=median(median(submatrix));

% submatrix(1:4,1:4)=md;

% cA1(i:i+4, j:j+4)=submatrix;

% end

% submatrix = cH1(i:i+4, j:j+4);

% entpy=entropyfilt(submatrix);

% if (entpy<std2(submatrix)\*sqrt(2\*log(sum(sum(submatrix>0)))))

% md=median(median(submatrix));

% submatrix(1:4,1:4)=md;

% cH1(i:i+4, j:j+4)=submatrix;

% end

% submatrix = cV1(i:i+4, j:j+4);

% entpy=entropyfilt(submatrix);

% if (entpy<std2(submatrix)\*sqrt(2\*log(sum(sum(submatrix>0)))))

% md=median(median(submatrix));

% submatrix(1:4,1:4)=md;

% cV1(i:i+4, j:j+4)=submatrix;

% end

% submatrix = cD1(i:i+4, j:j+4);

% entpy=entropyfilt(submatrix);

% if (entpy<std2(submatrix)\*sqrt(2\*log(sum(sum(submatrix>0)))))

% md=median(median(submatrix));

% submatrix(1:4,1:4)=md;

% cD1(i:i+4, j:j+4)=submatrix;

% end

% j=j+4;

% end

% i=i+4;

% end

% figure,imshow(cA1),title('Thresholding Compresseion at level 1');

%% Second level compression

cA1=im2double(mat2gray(cA1));

[cA2,cH2,cV2,cD2] = dwt2(cA1,'sym4');

figure,

subplot(2,2,1);imshow(cA2);title('CA2');

subplot(2,2,2);imshow(cH2);title('CH2');

subplot(2,2,3);imshow(cV2);title('CV2');

subplot(2,2,4);imshow(cD2);title('CD2');

%% Thresholding at level 2

[x,y]=size(cA2);

disp(x);

disp(y);

for i=1:x-4

for j=1: y-4

submatrix = cA2(i:i+4, j:j+4);

entpy=entropyfilt(submatrix);

if (entpy<std2(submatrix)\*sqrt(2\*log(sum(sum(submatrix>0)))))

md=median(median(submatrix));

submatrix(1:4,1:4)=md;

cA2(i:i+4, j:j+4)=submatrix;

end

submatrix = cH2(i:i+4, j:j+4);

entpy=entropyfilt(submatrix);

if (entpy<std2(submatrix)\*sqrt(2\*log(sum(sum(submatrix>0)))))

md=median(median(submatrix));

submatrix(1:4,1:4)=md;

cH2(i:i+4, j:j+4)=submatrix;

end

submatrix = cV2(i:i+4, j:j+4);

entpy=entropyfilt(submatrix);

if (entpy<std2(submatrix)\*sqrt(2\*log(sum(sum(submatrix>0)))))

md=median(median(submatrix));

submatrix(1:4,1:4)=md;

cV2(i:i+4, j:j+4)=submatrix;

end

submatrix = cD2(i:i+4, j:j+4);

entpy=entropyfilt(submatrix);

if (entpy<std2(submatrix)\*sqrt(2\*log(sum(sum(submatrix>0)))))

md=median(median(submatrix));

submatrix(1:4,1:4)=md;

cD2(i:i+4, j:j+4)=submatrix;

end

j=j+4;

end

i=i+4;

end

figure,imshow(cA2),title('Thresholding Compresseion at DWT levl 2');

%% checking

X2=idwt2(cA2,cH2,cV2,cD2,'sym4');

figure,imshow(X2),title('Check Reconstuceted from CA2');

%% preparing for second stage compression

I=cA2;

I=I(1:128,1:128);%resize image into 256\*256

figure,imshow(I);title('input Image for quadtree compression');drawnow;

tic;%record time

% I=mat2gray(I);

% I=uint8(I\*255);

I = (I - min(I(:))) / (max(I(:)) - min(I(:)));

I = im2uint8(I);

%% 2.Quadtree Decomposition

s=qtdecomp(I,0.2,[2 64]);%divides image using quadtree decomposition of threshold .2 and min dim =2 ,max dim =64

[i,j,blksz] = find(s);

blkcount=length(i);

avg=zeros(blkcount,1);

for k=1:blkcount

avg(k)=mean2(I(i(k):i(k)+blksz(k)-1,j(k):j(k)+blksz(k)-1)); %value

end

avg=uint8(avg);

figure,imshow((full(s)));title('Quadtree Decomposition');drawnow;

%% 3.Huffman Encoding

i(end+1)=0;j(end+1)=0;blksz(end+1)=0;%set boundary elements

data=[i;j;blksz;avg];

data=single(data);

symbols= unique(data);

counts = hist(data(:), symbols);

p = counts./ sum(counts);% Probability distribution

sp=round(p\*1000);

dict = huffmandict(symbols,p');

comp = huffmanenco(data,dict);% Encode the data.

%% 4.Compressed

t=toc;

fprintf('Time taken for compression = %f seconds\n',t);

%compression ratio

bits\_in\_original=8\*495\*495;

bits\_in\_final=length(comp)+8\*length(symbols)+8\*length(sp);

CR= bits\_in\_original/bits\_in\_final;

fprintf('compression ratio= %f\n',CR);

%% 5.Huffman Decoding

tic;

datanew = huffmandeco(comp,dict);

zeroindx=find(data==0);

inew=datanew(1:zeroindx(1)-1);

jnew=datanew(zeroindx(1)+1:zeroindx(2)-1);

blksznew=datanew(zeroindx(2)+1:zeroindx(3)-1);

avgnew=datanew(zeroindx(3)+1:end);

%% 6.Decompressed image

avgnew=uint8(avgnew);

for k=1:blkcount

outim(inew(k):inew(k)+blksznew(k)-1,jnew(k):jnew(k)+blksznew(k)-1)=avgnew(k);

end

figure,imshow(outim);title('Decompressed Image from Quadtree');drawnow;

%% PSNR calculation

%Time taken for De-compression

t=toc;

fprintf('Time taken for Decompression = %f seconds\n',t);

hpsnr = vision.PSNR;

psnr = step(hpsnr, I,outim);

fprintf('PSNR= %f\n',psnr);

outim2=im2double(outim);

X=idwt2(outim2,cH2,cV2,cD2,'sym4');

X2=idwt2(X,cH1,cV1,cD1,'sym4');

figure,imshow(X),title('Decompressed at 2');

figure,imshow(X2),title('Decompressed at 1');

%[cA1,cH1,cV1,cD1] = dwt2(i1,'sym4');