% Write and execute the MATLAB Code to learn the Huffman coding based

% image compression.

clear all;

clc;

Icolor = imread('G:\YCCE-27-11-2021\DIP-2021-2022\DIP-PR\Lenna.tif');

% (i)   Convert the given image to grey level image and visualize the

% obtained image.

Igrey = rgb2gray(Icolor);

SZIgrey = size(Igrey);

Rowpixels = SZIgrey(1);

Columnpixels = SZIgrey(2);

% figure(1), imshow(Igrey);

% (ii)  Generate the histogram of the obtained grey level image.

[counts,bins] = imhist(Igrey);

% figure(2), imhist(Igrey);

% (iii) Use the histogram data and generate the Huffman code for

% the grey level image.

greylevels = bins;

probabilities = counts./(Rowpixels\*Columnpixels);

% Transpose of greylevels & probabilities

symbols = greylevels.';

p = probabilities.';

% HUFFMAN ENCODING

CompStartTime = tic;

[dict,avglen] = huffmandict(symbols,p);

IgreyColumnVector = Igrey(:);

code = huffmanenco(IgreyColumnVector,dict);

CompEndTime = toc(CompStartTime);

% (iv)  Evaluate the compression time, bits per pixel, and

% compression ratio.

CompressionTime = CompEndTime;

BitsPerPixel = avglen;

CompressionRatio = (Rowpixels\*Columnpixels\*8)/(Rowpixels\*Columnpixels\*avglen);

fprintf('\n The Compression Time is %0.4f', CompressionTime);

fprintf('\n The Bits Per Pixel is %0.4f', BitsPerPixel);

fprintf('\n The Compression Ratio is %0.4f', CompressionRatio);

% (v)   Decompress the encoded data and reconstruct the image.

DeCompStartTime = tic;

sig = huffmandeco(code,dict);

DeCompEndTime = toc(DeCompStartTime);

IIgrey = reshape(sig,Rowpixels,[]);

isequal(Igrey,IIgrey);

if ans == 1

fprintf('\n Both the Original and Reconstructed Images are SAME');

end

% (vi)  Evaluate the decompression time and reconstructed

% image quality through PSNR.

DeCompressionTime = DeCompEndTime;

[peaksnr, snr] = psnr(uint8(IIgrey), Igrey);

fprintf('\n The Decompression Time is %0.4f', DeCompressionTime);

fprintf('\n The Peak-SNR value is %0.4f', peaksnr);

fprintf('\n The SNR value is %0.4f \n', snr);

% (vii) Plot the results obtained (Original grey level image and

% reconstructed image with the display of compression time,

% bits per pixel, compression ratio, decompression time, and PSNR).

figure;

subplot(2,4,1);

imshow(Igrey);

title('Original GL Image');

subplot(2,4,5);

imshow(uint8(IIgrey));

title('Reconstructed GL Image');

subplot(2,4,2);

axis off;

str = {'Comp.Time is',CompressionTime'};

text(0,0.6,str);

subplot(2,4,3);

axis off;

str = {'BPP', BitsPerPixel};

text(0,0.6,str);

subplot(2,4,4);

axis off;

str = {'CR is', CompressionRatio};

text(0,0.6,str);

subplot(2,4,6);

axis off;

str = {'Decom.Time is', DeCompressionTime};

text(0,0.6,str);

subplot(2,4,7);

axis off;

str = {'PSNR  is', peaksnr};

text(0,0.6,str);

subplot(2,4,8);

axis off;

str = {'The SNR value is', snr};

text(0,0.6,str);

% Write and execute the MATLAB Code to learn the LZW image

% compression technique.

% (i) Convert the given image to grey level image and visualize the

% obtained image.

clear all;

clc;

% % I = [39 39 126 126; 39 39 126 126; 39 39 126 126; 39 39 126 126];

% % I = uint8(I);

% % Igrey = I;

Icolor = imread("/MATLAB Drive/Lenna.tiff");

I = rgb2gray(Icolor);

Igrey = I;

SZIgrey = size(Igrey);

Rowpixels = SZIgrey(1);

Columnpixels = SZIgrey(2);

Igrey = Igrey';

IIgrey = Igrey(:);

IIgrey = IIgrey';

str = IIgrey;

% figure(1), imshow(Igrey)

% (ii) Apply the LZW compression algorithm for the grey level image.

CompStartTime = tic;

[EncodedData, CodeCount, table]=lzwCompression(uint8(str));

CompEndTime = toc(CompStartTime);

EncodedOutput = EncodedData-1;

% (iii) Evaluate the compression time, bits per pixel, and compression ratio.

CompressionTime = CompEndTime;

DictLocation = table.nextCode;

FinalDictLocation = DictLocation - 1;

bits = ceil(log2(FinalDictLocation+1));

avglen = (CodeCount\*bits)/(Rowpixels\*Columnpixels);

BitsPerPixel = avglen;

CompressionRatio = (Rowpixels\*Columnpixels\*8)/(Rowpixels\*Columnpixels\*avglen);

fprintf('\n The Compression Time is %0.4f', CompressionTime);

fprintf('\n The Bits Per Pixel is %0.4f', BitsPerPixel);

fprintf('\n The Compression Ratio is %0.4f', CompressionRatio);

% (iv) Decompress the encoded data and reconstruct the image.

DeCompStartTime = tic;

[DecodedData,table]=lzwDecompression(EncodedData);

DeCompEndTime = toc(DeCompStartTime);

IRgrey = reshape(DecodedData,Rowpixels,[]);

IRgrey = IRgrey';

isequal(I,IRgrey);

if ans == 1

fprintf('\n Both the Original and Reconstructed Images are SAME');

end

% % transfor it back to char array

% unpacked = char(unpacked);

% % test

% isOK = strcmp(str,unpacked)

% % show new table elements

Z=table.codes(257:end);

% Y=char(Z);

% char(table.codes{257:end})

% (v) Evaluate the decompression time and reconstructed image quality through PSNR.

DeCompressionTime = DeCompEndTime;

[peaksnr, snr] = psnr(uint8(IRgrey), I);

fprintf('\n The Decompression Time is %0.4f', DeCompressionTime);

fprintf('\n The Peak-SNR value is %0.4f', peaksnr);

fprintf('\n The SNR value is %0.4f \n', snr);

% (vi) Plot the results obtained (Original grey level image and reconstructed image with the display of compression time, bits per pixel, compression ratio, decompression time, and PSNR).

%

figure;

subplot(2,4,1);

imshow(I);

title('Original GL Image');

subplot(2,4,5);

imshow(uint8(IRgrey));

title('Reconstructed GL Image');

subplot(2,4,2);

axis off;

str = {'Comp.Time is',CompressionTime'};

text(0,0.6,str);

subplot(2,4,3);

axis off;

str = {'BPP', BitsPerPixel};

text(0,0.6,str);

subplot(2,4,4);

axis off;

str = {'CR is', CompressionRatio};

text(0,0.6,str);

subplot(2,4,6);

axis off;

str = {'Decom.Time is', DeCompressionTime};

text(0,0.6,str);

subplot(2,4,7);

axis off;

str = {'PSNR is', peaksnr};

text(0,0.6,str);

subplot(2,4,8);

axis off;

str = {'The SNR value is', snr};

text(0,0.6,str);

clc;

close all;

clear all;

% Read Colour Image and convert it to a grey level Image

% Display the original Image

mycolourimage = imread("/MATLAB Drive/Lenna.tiff");

myimage = rgb2gray(mycolourimage);

subplot(3,3,1);

imshow(myimage); title('Original Image');

% Apply Sobel Operator

% Display both horizontal and vertical Edges

sobelvrthz = edge(myimage,'sobel','both');

subplot(3,3,2);

imshow(sobelvrthz,[]); title('Sobel - All edges');

% Apply Roberts Operator

% Display both horizontal and vertical Edges

robertsedg = edge(myimage,'roberts');

subplot(3,3,3);

imshow(robertsedg,[]); title('Roberts - Edges');

% Apply Prewitt Operator

% Display both horizontal and vertical Edges

robertsedg = edge(myimage,'prewitt');

subplot(3,3,4);

imshow(robertsedg,[]); title('Prewitt - Edges');

% Apply Canny edge detection

cannyedg = edge(myimage,'canny');

subplot(3,3,5);

imshow(cannyedg,[]); title('Canny Edge');

% Write and execute the MATLAB Code to learn the region growing

% based image segmentation.

clear all;

clc;

% Icolor = imread('G:\YCCE-27-11-2021\DIP-2021-2022\DIP-PR\Lenna.tif');

Icolor = imread("/MATLAB Drive/Lenna.tiff");

% (i) Convert the given image to grey level image and visualize the

% obtained image.

Igrey = rgb2gray(Icolor);

SZIgrey = size(Igrey);

Rowpixels = SZIgrey(1);

Columnpixels = SZIgrey(2);

I = Igrey;

% figure(1), imshow(Igrey);

% (ii) Generate the histogram of the obtained grey level image.

[counts,bins] = imhist(I);

% figure(2), imhist(I);

S = 255; % Single seed value

T = 65; % T is a scalar value

% rgis function call

[IO, NR , SI , TI ] = rgis(I, S, T);

% (iv) Plot the results obtained ((a) Original grey level image,

% (b) histogram, (c) Seed points image, (d) Binary image showing all

% the pixels (in white) that passed the threshold test, and

% (e) Result after all the pixels in (d) were analyzed for

% 8-connectivity to the seed points).

figure;

subplot(2,3,1);

imshow(Icolor);

title('Original Color Image');

subplot(2,3,2);

imshow(uint8(I));

title('Original GL Image');

subplot(2,3,3);

imhist(I);

title('Histogram of GL Image');

subplot(2,3,4);

imshow(SI);

title('Seed Points Image');

subplot(2,3,5);

imshow(TI);

title('Binary Image Passed T.T.');

subplot(2,3,6);

imshow(IO);

title('Segmented Image');

% Write and execute the MATLAB Code to learn the 4-directional chain code and 8-directional chain code.

% Refer the noisyimage.jpg image and perform the following:

clear all;

clc;

Icolor = imread("/MATLAB Drive/prac10/Orignal Noisy Image/noisyimage.jpg");

% (i) Convert the given image to grey level image and visualize the

% obtained image.

Igrey = rgb2gray(Icolor);

SZIgrey = size(Igrey);

Rowpixels = SZIgrey(1);

Columnpixels = SZIgrey(2);

% figure(1), imshow(Igrey);

% (i) Smooth the given image by using 9×9 averaging mask.

h = fspecial('average', 9);

g = imfilter(Igrey, h, 'replicate');

zz = g;

% figure(2), imshow(g);

% (ii) Generate the thresholded image from the result obtained in (i).

gB = im2bw(g, 0.5);

B = bwboundaries(gB, 'noholes');

d = cellfun('length', B) ;

[maxd, k] = max(d);

b = B{k};

% figure(3), imshow(gB);

% (iii) Generate the boundary image from the result obtained in (ii).

[M N] = size(g);

g = bound2im(b, M, N);

% figure(4), imshow(g);

% (iv) Generate the subsampled boundary image from the result obtained in (iii).

[s, su] = bsubsamp(b, 50);

g2 = bound2im(s, M, N);

% figure(5), imshow(g2);

% (v) Generate the connected points image from the result obtained in (iv).

cn = connectpoly( s(:, 1), s(:, 2));

g3 = bound2im(cn, M, N);

% figure(6), imshow(g3);

% (vi) Write the functions for (i), (ii), (iii), (iv), (v) and then use

% these functions to write the generalized code for the evaluation of

% chain codes (4-directional and 8-directional.

c = fchcode(su);

% (vii) Plot the results obtained ((a) Original noisy image,

% (b) result obtained in (i), (c) result obtained in (ii),

% (d) result obtained in (iii), (e) result obtained in (iv), and

% (f) result obtained in (v)).

figure(1), imshow(Igrey);

title('Original noisy image');

f = gcf;

exportgraphics(f,'1.png','Resolution',300)

figure(2), imshow(zz);

title('Smoothed Image');

f = gcf;

exportgraphics(f,'2.png','Resolution',300)

figure(3), imshow(gB);

title('Thresholded Image');

f = gcf;

exportgraphics(f,'3.png','Resolution',300)

figure(4), imshow(g);

title('Boundary of Binary Image');

f = gcf;

exportgraphics(f,'4.png','Resolution',300)

figure(5), imshow(g2);

title('Subsampled Boundary');

f = gcf;

exportgraphics(f,'5.png','Resolution',300)

figure(6), imshow(g3);

title('Connected Points');

f = gcf;

exportgraphics(f,'6.png','Resolution',300)

% (viii) Display the generated 4-directional chain code and 8-directional chain code.

% TO DISPLAY THE RESULTS ABOUT CHAIN CODE AND DIFFERENCE

% JUST TYPE c ON COMMAND PROMPT

% THEN TAKE SCREEN SHOT AS OUTPUT

% RESULT WILL BE DISPLAYED ON COMMAND PROMPT AS FOLLOWS:

% x0y0: [4 2]

% fcc: [2 2 0 2 0 0 6 0 6 4 6 0 6 4 6 4 4 4 2 2]

% diff: [0 6 2 6 0 6 2 6 6 2 2 6 6 2 6 0 0 6 0 0]

% mm: [0 0 6 0 6 4 6 0 6 4 6 4 4 4 2 2 2 2 0 2]

% diffmm: [0 6 2 6 6 2 2 6 6 2 6 0 0 6 0 0 0 6 2 6]