Ece442

Lab #2

Image Representation

Linjian Xiang (1373037)

Questions:

1. Most of flowers and butterflies as well as some maple leave have strong red.

The background has stronger green.

Most part of figure has low blue, except the left top flower and a few flowers in the center of picture have stronger blue.

     2.  The average value of B2 is 71.9486

     3.  Pink colors became purple and green color parts got some blueish.

     4. Those images are very similar, we can observe that YCBCR images can be restored very well even they are compressed by a large ratio. In this way, while transfer images, we can change RGB images into YCBCR images and send after compressing. The image size got smaller and quality would not decrease a lot after restoring image.

Matlab code is shown in Appendix.

    5. Matlab code is shown in Appendix.

    6. The colour halftoned image has not bad image quality, it’s clear to tell color intensity and contrast. The image is combined with white, red, blue, and green pixels.

       Matlab code is shown in Appendix.

Appendix:

Q4:

%% Q4 YCbCr

peppers = imread('peppers.png');

[RGB\_2 ,RGB\_4 ,RGB\_8]=down\_up\_sample(peppers);

%% Q4 function

function [RGB\_2 RGB\_4 RGB\_8]=down\_up\_sample(Im)

% Convert RGB version of Im to an YCbCr version and seperate each component

Im\_Ycbcr = rgb2ycbcr(Im);

Y = Im\_Ycbcr(:,:,1);

Cb = Im\_Ycbcr(:,:,2);

Cr = Im\_Ycbcr(:,:,3);

% Chrominance downsampling by 2, 'bicubic'

Cb\_2 = imresize(Cb,0.5,'bicubic');

Cr\_2 = imresize(Cr,0.5,'bicubic');

% Chrominance upsampling by 2, 'bicubic'

Cb\_2 = imresize(Cb\_2,2,'bicubic');

Cr\_2 = imresize(Cr\_2,2,'bicubic');

size(Cr\_2)

% Create new YCbCr Image and convert back to RGB

[height width depth]=size(Im\_Ycbcr);

Im\_Ycbcr\_2 = zeros(height,width,depth);

Im\_Ycbcr\_2(:,:,1) = double(Y);

Im\_Ycbcr\_2(:,:,2) = double(Cb\_2);

Im\_Ycbcr\_2(:,:,3) = double(Cr\_2);

Im\_Ycbcr\_2 = uint8(Im\_Ycbcr\_2);

RGB\_2= ycbcr2rgb(Im\_Ycbcr\_2);%save result to RGB\_2

% Chrominance downsampling by 4, 'bicubic'

Cb\_4 = imresize(Cb,0.25,'bicubic');

Cr\_4 = imresize(Cr,0.25,'bicubic');

% Chrominance upsampling by 4, 'bicubic'

Cb\_4 = imresize(Cb\_4,4,'bicubic');

Cr\_4 = imresize(Cr\_4,4,'bicubic');

% Create new YCbCr Image and convert back to RGB

Im\_Ycbcr\_4 = zeros(height,width,depth);

Im\_Ycbcr\_4(:,:,1) = double(Y);

Im\_Ycbcr\_4(:,:,2) = double(Cb\_4);

Im\_Ycbcr\_4(:,:,3) = double(Cr\_4);

Im\_Ycbcr\_4 = uint8(Im\_Ycbcr\_4);

RGB\_4= ycbcr2rgb(Im\_Ycbcr\_4);%save result to RGB\_4

% Chrominance downsampling by 8, 'bicubic'

Cb\_8 = imresize(Cb,0.125,'bicubic');

Cr\_8 = imresize(Cr,0.125,'bicubic');

% Chrominance upsampling by 8, 'bicubic'

Cb\_8 = imresize(Cb\_8,8,'bicubic');

Cr\_8 = imresize(Cr\_8,8,'bicubic');

% Create new YCbCr Image and convert back to RGB

Im\_Ycbcr\_8 = zeros(height,width,depth);

Im\_Ycbcr\_8(:,:,1) = double(Y);

Im\_Ycbcr\_8(:,:,2) = double(Cb\_8);

Im\_Ycbcr\_8(:,:,3) = double(Cr\_8);

Im\_Ycbcr\_8 = uint8(Im\_Ycbcr\_8);

RGB\_8= ycbcr2rgb(Im\_Ycbcr\_8);%save result to RGB\_8

%image write

imwrite(RGB\_2, 'paper\_sampling\_2.png');

imwrite(RGB\_4, 'paper\_sampling\_4.png');

imwrite(RGB\_8, 'paper\_sampling\_8.png');

% Display the result

figure()

subplot(2,2,1),imshow(Im),title('Original Image');

subplot(2,2,2),imshow(RGB\_2),title('Sampling factor=2');

subplot(2,2,3),imshow(RGB\_4),title('Sampling factor=4');

subplot(2,2,4),imshow(RGB\_8),title('Sampling factor=8');

Q5:

%% Part 2, Image halftoning test

lena = imread('lenna.bmp');

lena\_halftone=myhalftone(lena);

imwrite(lena\_halftone, 'lenna\_halftone.bmp');

%Q5 function

function Im2=myhalftone(Im)

close all

Im\_double=double(Im); % convert to double precision before processing

[rows,cols] = size(Im\_double); % # of rows and columns

Im2 = zeros(rows,cols); % initialization of the binary image

Im\_double2 = zeros(rows+2, cols+2);

Im\_double2(2:rows+1,2:cols+1) = Im\_double;

for i = 1 : rows % processing the pixels in the raster scan order

for j = 1 : cols

% set Im2(i,j) to the appropriate binary value based on Im\_double

if Im\_double2(i+1,j+1) < 128

Im2(i,j) = 0;

else

Im2(i,j) = 255;

end

% compute the quantization error

e = Im\_double2(i+1,j+1) - Im2(i,j);

% diffuse the quantization error to neighboring pixels of Im\_double

Im\_double2(i+1,j+2) = Im\_double2(i+1,j+2) + e \* 7/16;

Im\_double2(i+2,j) = Im\_double2(i+2,j) + e \* 3/16;

Im\_double2(i+2,j+1) = Im\_double2(i+2,j+1) + e \* 5/16;

Im\_double2(i+2,j+2) = Im\_double2(i+2,j+2) + e \* 1/16;

end

end

Im2=uint8(Im2); %cast back to unsigned 8-bit integers

% display the original image

figure(1)

imshow(Im);

% display the binary image

figure(2)

imshow(Im2); title('halftoned lean');

end

Q6:

%% Colour halftoning

Im=imread('lenna.png');

Im2=colourhalftone(Im);

imwrite(Im2, 'lenna\_colourhalftone.png');

%Q6 function

function Im2=colourhalftone(Im)

[rows ,cols ,depth]=size(Im);

Im2=zeros(rows,cols,depth,'uint8');

% Extract C,M, and Y components from Im

R = Im(:,:,1); G = Im(:,:,2); B = Im(:,:,3);

C = 255 - R; M = 255 - G; Y = 255 - B;

% Process each component using Floyd-Steinberg method

C\_new=myhalftone(C);

M\_new=myhalftone(M);

Y\_new=myhalftone(Y);

% Convert C\_new, M\_new, and Y\_new back to RGB and save to Im2

R\_new = 255 - C\_new;

G\_new = 255 - M\_new;

B\_new = 255 - Y\_new;

Im2(:,:,1) = (R\_new);

Im2(:,:,2) = (G\_new);

Im2(:,:,3) = (B\_new);

% display the original image

figure(1)

imshow(Im);

% display the binary image

figure(2)

imshow(Im2); title('colour halftoned lenna');

end