close all;

clear all;

clc;

% Acquiring the image from folder

im = imread('F:\lemon images\ripe\yellow1.jpg');

% RESIZING THE IMAGES

[~, ~, ~] = size(im);

p = imresize(im, [250, 250]);

figure(1),

subplot(2,3,1), imshow(p), title('Original Image');

% Converting the image from RGB to CMY

C = 1 - double(p(:,:,1)) / 255;

M = 1 - double(p(:,:,2)) / 255;

Y = 1 - double(p(:,:,3)) / 255;

% Combining the CMY channels into one image

CMY = cat(3, C, M, Y);

% Display the CMY image

subplot(2,3,2), imshow(CMY), title('CMY Image');

% Convert CMY to grayscale for edge detection

I = rgb2gray(im2uint8(CMY));

% Apply edge detection

[~, threshold] = edge(I, 'sobel');

fudgeFactor = 0.5;

BWs = edge(I, 'sobel', threshold \* fudgeFactor);

subplot(2,3,3), imshow(BWs), title('Binary Gradient Mask');

% Fill the holes

BWdfill = imfill(BWs, 'holes');

subplot(2,3,4), imshow(BWdfill), title({'Binary Image with', 'Filled Holes'});

% Filter the image

fim = medfilt2(BWdfill);

subplot(2,3,5), imshow(fim), title({'Filtered and', 'Segmented Image'});

% Converting the segmented image to color image

[r, c] = size(fim);

colimg = uint8(zeros(r, c, 3));

for i = 1:r

for j = 1:c

if fim(i, j) == 0

colimg(i, j, :) = uint8(fim(i, j));

else

colimg(i, j, :) = uint8(p(i, j, :));

end

end

end

subplot(2,3,6), imshow(colimg), title('Color Image');

I1 = colimg;

grim = rgb2gray(I1);

c1 = 0;

[r, c] = size(grim);

for i = 1:r

for j = 1:c

if grim(i, j) > 0

c1 = c1 + 1;

end

end

end

% Extract CMY channels

cyan\_channel = colimg(:, :, 1);

magenta\_channel = colimg(:, :, 2);

yellow\_channel = colimg(:, :, 3);

% Calculate histograms for each channel

num\_bins = 256;

cyan\_hist = imhist(cyan\_channel, num\_bins);

magenta\_hist = imhist(magenta\_channel, num\_bins);

yellow\_hist = imhist(yellow\_channel, num\_bins);

% Normalize the histograms

cyan\_hist = cyan\_hist / sum(cyan\_hist);

magenta\_hist = magenta\_hist / sum(magenta\_hist);

yellow\_hist = yellow\_hist / sum(yellow\_hist);

% Plot histograms

figure;

subplot(1, 3, 1);

bar(0:num\_bins-1, cyan\_hist, 'c');

title('Cyan Channel Histogram');

xlabel('Intensity');

ylabel('Frequency');

subplot(1, 3, 2);

bar(0:num\_bins-1, magenta\_hist, 'm');

title('Magenta Channel Histogram');

xlabel('Intensity');

ylabel('Frequency');

subplot(1, 3, 3);

bar(0:num\_bins-1, yellow\_hist, 'y');

title('Yellow Channel Histogram');

xlabel('Intensity');

ylabel('Frequency');

% Calculate the mean value for each channel

mean\_cyan = mean(cyan\_channel(:));

mean\_magenta = mean(magenta\_channel(:));

mean\_yellow = mean(yellow\_channel(:));

% Display the mean values

%fprintf('Mean value of CMY color based on Histogram Mean\n');

%fprintf('Mean value of Cyan channel: %.2f\n', mean\_cyan);

%fprintf('Mean value of Magenta channel: %.2f\n', mean\_magenta);

%fprintf('Mean value of Yellow channel: %.2f\n', mean\_yellow);

imC = colimg;

imC(:,:,2:3) = 0;

imM = I1;

imM(:,:,1:2:3) = 0;

imY = I1;

imY(:,:,1:2) = 0;

C\_val = ((sum(imC(:))) / c1);

M\_val = ((sum(imM(:))) / c1);

Y\_val = ((sum(imY(:))) / c1);

% Feature Extraction

GI = rgb2gray(I1);

glcm = graycomatrix(GI);

R1 = graycoprops(glcm);

fprintf('Mean value of CMY color based on greenpoints\n');

fprintf('Cmean\tMmean\tYmean\n');

fprintf('%f\t%f\t%f\n', C\_val, M\_val, Y\_val);

fprintf('\n........SEGMENTATION THROUGH COLOUR VALUES........\n\n');

greenpoints = colimg(:,:,1) >= 125 & colimg(:,:,2) <= 255 & colimg(:,:,3) <= 100;

percentgreen = 100 \* (sum(sum(greenpoints)) / (size(colimg, 1) \* size(colimg, 2)));

fprintf('image has %d cyan pixels\n\n', sum(sum(greenpoints)));

fprintf('image is %.2f percent cyan\n\n', percentgreen);

% Displaying output based on threshold value

if (percentgreen > 0 && percentgreen < 6)

fprintf('Lemon is tender\n');

elseif (percentgreen > 6 && percentgreen < 16)

fprintf('Lemon is semi-ripe\n');

elseif (percentgreen > 16 && percentgreen < 30)

fprintf('Lemon is ripe\n');

end

**// USING RGB COLOR MODEL //**

close all;

clear all;

clc;

%Acquiring the image from folder

im = imread('C:\Users\User\Desktop\green8.jpg');

%RESIZING THE IMAGES

[~, ~, ~] = size(im);

p=imresize(im,[250,250]);

figure(1),

subplot(2,3,1),imshow(p),title('original image');

%Converting the original image to gray scale

I = rgb2gray(p);

subplot(2,3,2),imshow(I),title('Gray Image')

%applying binary gradient Ma

[~,threshold] = edge(I,'sobel');

fudgeFactor = 0.5

BWs = edge(I,'sobel',threshold \* fudgeFactor);

subplot(2,3,3),imshow(BWs),title('Binary Gradient Mask');

% to fill the holes

BWdfill = imfill(BWs,'holes');

subplot(2,3,4),imshow(BWdfill)

title({'Binary Image with','Filled Holes'});

% to filter the image

[r,c]=size(BWdfill);

fim=medfilt2(BWdfill);

subplot(2,3,5),imshow(fim),title({'Filtered and',' Segmented Image'});

% converting the segmented image to color image

colimg=uint8(zeros(r,c));

for i=1:r

for j=1:c

for k=1:3

if fim(i,j)==0

colimg(i,j,k)=uint8(fim(i,j));

else

colimg(i,j,k)=uint8(p(i,j,k));

end

end

end

end

subplot(2,3,6),imshow(colimg),title('color image');

I1=colimg;

grim=rgb2gray(I1);

c1=0;

[r,c]=size(grim);

for i=1:r

for j=1:c

if(grim(i,j)>0)

c1=c1+1;

end

end

end

% Extract RGB channels

red\_channel = colimg(:, :, 1);

green\_channel = colimg(:, :, 2);

blue\_channel = colimg(:, :, 3);

% Calculate histograms for each channel

num\_bins = 256;

red\_hist = imhist(red\_channel, num\_bins);

green\_hist = imhist(green\_channel, num\_bins);

blue\_hist = imhist(blue\_channel, num\_bins);

% Normalize the histograms

red\_hist = red\_hist / sum(red\_hist);

green\_hist = green\_hist / sum(green\_hist);

blue\_hist = blue\_hist / sum(blue\_hist);

% Plot histograms

figure;

subplot(1, 3, 1);

bar(0:num\_bins-1, red\_hist, 'r');

title('Red Channel Histogram');

xlabel('Intensity');

ylabel('Frequency');

subplot(1, 3, 2);

bar(0:num\_bins-1, green\_hist, 'g');

title('Green Channel Histogram');

xlabel('Intensity');

ylabel('Frequency');

subplot(1, 3, 3);

bar(0:num\_bins-1, blue\_hist, 'b');

title('Blue Channel Histogram');

xlabel('Intensity');

ylabel('Frequency');

% Calculate the mean value for each channel

mean\_red = mean(red\_channel(:));

mean\_green = mean(green\_channel(:));

mean\_blue = mean(blue\_channel(:));

% Display the mean values

%fprintf('Mean value of RGB color based on Histogram Mean\n')

%fprintf('Mean value of Red channel: %.2f\n', mean\_red);

%fprintf('Mean value of Green channel: %.2f\n', mean\_green);

%fprintf('Mean value of Blue channel: %.2f\n', mean\_blue);

imR=colimg;

imR(:,:,2:3)=0;

imG=I1;

imG(:,:,1:2:3)=0;

imB=I1;

imB(:,:,1:2)=0;

R(1)=((sum(imR(:)))/c1);

R(2)=mean(imR(:));

R(3)=var(double(imR(:)));

G(1)=((sum(imG(:)))/c1);

G(2)=std2(imG(:));

G(3)=var(double(imG(:)));

B(1)=((sum(imB(:)))/c1);

B(2)=std2(imB(:));

B(3)=var(double(imB(:)));

%Feature Extraction

GI=rgb2gray(I1);

glcm = graycomatrix(GI);

R1 = graycoprops(glcm);

%R2(2) = R1.Correlation;

%R2(3) = R1.Energy;

%R2(4) = R1.Homogeneity;

%R2(5)=entropy(GI);

fprintf('Mean value of RGB color\n')

fprintf('Rmean\tGmean\tBmean\n')

fprintf('%f\t%f\t%f\n',R(1),G(1),B(1));

%textureval = R1.Homogeneity\*10;

%disp TextureValue

%disp(textureval/10);

fprintf('\n........RESULT........\n\n');

greenpoints=colimg(:,:,1)>=125 & colimg(:,:,2)<=255 & colimg(:,:,3)<=100;

percentgreen=100\*(sum(sum(greenpoints))/(size(colimg,1)\*size(colimg,2)));

fprintf('image has %d green pixels\n\n',sum(sum(greenpoints)));

fprintf('image is %.2f percent green\n\n',percentgreen);

%displaying output based on threshold value

if (sum(sum(percentgreen))>0&&sum(sum(percentgreen))<6)

fprintf('lemon is TENDER\n');

end

if (sum(sum(percentgreen))>6&&sum(sum(percentgreen))<16)

fprintf(' lemon is SEMI-RIPE\n');

end

if (sum(sum(percentgreen))>16&&sum(sum(percentgreen))<30)

fprintf('lemon is RIPE\n');

end