Michael Hickey

Professor Tom

Digital Signals Processing

Lab 04 Greyscale Filters

Problem 1

Arithmetic Greyscale

image = arithmetic\_greyscale(man);

imshow(image)

function [ newPhoto ] = arithmetic\_greyscale( photo )

%get the rows, columns, and color chanels of a photo

[rows, columns, chanels] = size(photo);

%chanels is currently not used.

%%Get the dimensions of the old photo and make the new photo the same

%%dimensions

newPhoto = photo;

for rowsPixel = 1:rows

for columnsPixel = 1:columns

%if the color is red

R = photo(rowsPixel, columnsPixel, 1);

%if the color is green

G = photo(rowsPixel, columnsPixel, 2);

%if the color is blue

B = photo(rowsPixel, columnsPixel, 3);

%average all the RGB colors

%the number 3 is the number of chanels

averageColor = (R + G + B)/3;

%Replace every RGB color value with the averageColor value

newPhoto(rowsPixel, columnsPixel, 1) = averageColor;

newPhoto(rowsPixel, columnsPixel, 2) = averageColor;

newPhoto(rowsPixel, columnsPixel, 3) = averageColor;

end

end

end

Problem 2

Geometric Greyscale

image = geometric\_greyscale(man);

imshow(image)

function [ new\_photo ] = geometric\_greyscale( photo1 )

%get the rows, columns, and color chanels of a photo

[rows, columns, chanels] = size(photo1);

%currently chanels is not used.

%%Photo is a double

photo = im2double(photo1);

%%Get the dimensions of the old photo and make the new photo the same

%%dimensions

new\_photo = photo;

for rows\_pixels = 1:rows

for columns\_pixels = 1:columns

%if the color is red

R = photo(rows\_pixels, columns\_pixels, 1);

%if the color is green

G = photo(rows\_pixels, columns\_pixels, 2);

%if the color is blue

B = photo(rows\_pixels, columns\_pixels, 3);

%lets square all the color values, datatype = double

squared\_color\_values = R.^2 + G.^2 + B.^2;

%arithmetic defaults to a datatype = double

%this constant number is from the formula

constant\_number = 3 \* 255.^2;

%this is the numerator being inserted into the square root

sqrt\_of\_color\_values = sqrt(squared\_color\_values);

%this is the denominator being inserted into a square root

sqrt\_of\_constant = sqrt(constant\_number);

%performing the division part of the formula

%the result is a ratio of our total light levels.

average\_color = (sqrt\_of\_color\_values / sqrt\_of\_constant);

%We need to get the greyscale color, instead of the ratio

%Multiply the max light level by the ratio

average\_color = average\_color \* 255;

%%construct the new photo and for every color level, insert the

%%same average color.

%for the color red, make its value the average color

new\_photo(rows\_pixels, columns\_pixels, 1) = average\_color;

%for the color green, make its value the average color

new\_photo(rows\_pixels, columns\_pixels, 2) = average\_color;

%for the color blue make its value the average color

new\_photo(rows\_pixels, columns\_pixels, 3) = average\_color;

end

end

end

Problem 3.1

Arithmetic Greyscaled Result

A black and white photo of a person

Description generated with very high confidence

Geometric Greyscaled Result

A person standing on a baseball field

Description generated with high confidence

There are some significant differences between the two greyscaling processes.

The arithmetically greyscaled result looks very dark and has less detail. In comparison to the geometrically greyscaled result; the colors are smoother, and have more detail. There is much more definition in the geometrically greyscaled result, concerning contrast, contours, and average color, in comparison to the arithmetically greyscaled image.

The MatLab learning curve was quite steep when creating the geometrically greyscaling algorithm. I had to figure out what datatypes my variables were, and I needed to make sure every variable was properly formatted. Since there is no variable declaration type, Matlab will fit the variable to a datatype; I needed to use “whos” and a variable name to determine what datatypes my variables were. Matlab is completely different from C# or other languages.

Problem 3.2

We normalize our division by because we are using an image with three color channels which consist of 255 different values. The three color-channels (Reg, Green, and Blue) corresponds to the 3 in the formula. The 255 different color values (meaning how red is the pixels red value, how blue is the pixels blue value, and how green is the pixels green value) corresponds to the 255 in the formula. Next, the 255 is squared, because we need to apply a square root to comply with the geometric formula. Which consists of squaring the RGB values, adding them then using that sum as a radicand in a square root.

Problem 3.3

The geometric greyscale algorithm produced a better greyscaled image.

Problem 3.4

With my extensive knowledge of baseball (or google searching), I can deduce that the man in the photo is Dave Roberts. Dave Roberts steals second base by a split second in the ALCS 2004 playoff game. Then on the next batter’s hit, a line drive down past second base into the outfield, allowed for Roberts to take third base and continue onto home, scoring for the RedSox.

Here is a link to a video of the event on MLB’s website.

<https://www.mlb.com/news/dave-roberts-steal-set-amazing-2004-red-sox-playoff-run-in-motion/c-98844328>

Problem 4

Implement the Geometric Mean Color Emphasis

image2 = geometric\_mean\_color\_emphasis(WIT2, "yellow", 108.2);

imshow(image2)

function [ new\_photo ] = geometric\_mean\_color\_emphasis( photo1, color, threshold )

%This function will emphasize a color in a photo, and greyscale

%everything else

%color is a string, acceptable values are currently

%yellow

%get the rows, columns, and color chanels of a photo

[rows, columns, chanels] = size(photo1);

%currently chanels is not used.

%%Photo is a double

photo = im2double(photo1);

%%Get the dimensions of the old photo and make the new photo the same

%%dimensions

new\_photo = photo;

for rows\_pixels = 1:rows

for columns\_pixels = 1:columns

%if the color is red

R = photo(rows\_pixels, columns\_pixels, 1);

%if the color is green

G = photo(rows\_pixels, columns\_pixels, 2);

%if the color is blue

B = photo(rows\_pixels, columns\_pixels, 3);

%Defaulting the distance RGB color to 0

RGB = [0,0,0];

%If the parameter color is yellow, use the yellow color

if color == "yellow"

%WIT Yellow

RGB = [246, 222, 152];

%HEX code #f6de98

end

%Compute the distance between the specified RGB value and the

%actual pixel's RGB value

r\_dist = RGB(1) - R;

g\_dist = RGB(2) - G;

b\_dist = RGB(3) - B;

%Squaring each term

r\_dist2 = r\_dist^2;

g\_dist2 = g\_dist^2;

b\_dist2 = b\_dist^2;

%Sum the three terms

radicand = r\_dist2 + g\_dist2 + b\_dist2;

%Apply the square root to the sum. Lower the distance by 255

distance = sqrt(radicand) - 255;

%apply a threshold

if distance > threshold

%if the distance is greater than the threshold, we want to

%greyscale the pixel in that row & column

squared\_color\_values = R.^2 + G.^2 + B.^2;

constant\_number = 3 \* 255.^2;

sqrt\_of\_color\_values = sqrt(squared\_color\_values);

sqrt\_of\_constant = sqrt(constant\_number);

average\_color = (sqrt\_of\_color\_values / sqrt\_of\_constant);

average\_color = average\_color \* 255;

new\_photo(rows\_pixels, columns\_pixels, 1) = average\_color;

new\_photo(rows\_pixels, columns\_pixels, 2) = average\_color;

new\_photo(rows\_pixels, columns\_pixels, 3) = average\_color;

else

%continune to build the image without greyscaled pixels

new\_photo(rows\_pixels, columns\_pixels, 1) = R;

new\_photo(rows\_pixels, columns\_pixels, 2) = G;

new\_photo(rows\_pixels, columns\_pixels, 3) = B;

end

end

end

end

Geometric Mean Color Emphasis Greyscaled Result

A sign in front of a building

Description generated with high confidence

Original Image

A group of people in front of a building

Description generated with very high confidence

Problem 4.4

Not everything else was greyed out, some of the yellow grass stayed in the filtered image. Anything that was less than the threshold went through the filter.