Homework 5 – Color

Arya Araban

Abstract

Up until now we studied digital images that were either grayscale images, or if they were colored, we would convert them to become grayscale. In this article we explore how to work with colored images, as well as different types of color spaces, along with explaining their uses.

Report Info

Date: 1399/10/03

Keywords:
Colored Images
RGB Quantization
Color Spaces

means green 240 means blue. 60 degrees is yellow, 300 degrees is magenta)
Note that after finding the hue value we normalize it to be between [0,1].

The Saturation component signals how much the color is polluted with white color. The range of the S component is [0,1].

The Intensity range is between [0,1] where 0 means black, 1 means white.

In order to convert RGB to HSI, we do the following:

Steps to be followed:

1. Read a RGB image
2. Represent the RGB image in the range [0 1]
3. Find HSI components

$$\theta = \cos^{-1}\left\{\frac{\frac{1}{2}\left[(R-G)+(R-B)\right]}{\left[(R-G)^2+(R-B)(G-B)\right]^{1/2}}\right\}$$

4. H(Hue) = $\begin{cases}\theta & \text{If B<=G}\\ 360-\theta & \text{If B>G}\end{cases}$

5. S(Saturation) = 1- $\frac{3}{(R+G+B)}\left[\min(R,G,B)\right]$

6. I(Intensity)= $\frac{1}{3}(R+G+B)$

1- Introduction

There are different ways to represent a colored image. Each of these methods are called a "Color Space", with the most common color space being the RGB color space. The reason for this is that using the three main colors: red, green, blue, we can create most of the colors that are recognizable to the human eye. Other than RGB, we can also make use of other color spaces such as HSI, These color spaces allow us to have varying control over different aspects of an image.

2- Technical Description

Problem 5.1.1:

Convert an RGB image to HSI format, and discuss what each of the H, S, I components represent.

Solution:

The HSI color model represents every color with three components: hue (H), saturation (S), intensity (I).

The Hue component describes the color itself in the form of an angle between [0,360] degrees. (0 degrees mean red, 120

In the Munsell color space, a color is fully specified by listing the three numbers for hue, value, and chroma in that order.

We'll explain Each of three independent properties of this color space (which can be represented in three dimensions as an irregular color solid):

1-Hue:

Each horizontal circle Munsell divided into five principal hues: Red, Yellow, Green, Blue, and Purple, along with 5 intermediate hues (e.g., YR) halfway between adjacent principal hues.[4] Each of these 10 steps, with the named hue given number 5, is then broken into 10 sub-steps, so that 100 hues are given integer values.

The Munsell Hues, when the value and Chroma both equal 6

2-Value:

Value (also known as lightness) varies vertically along the color solid, from black (value 0) at the bottom, to white (value 10) at the top. Neutral grays lie along the vertical axis between black and white.

3-Chroma:

Chroma, measured radially from the center of each slice, represents the "purity" of a color (which is also related to saturation), with lower chroma being less pure (more washed out, as in pastels). Note that there is no upper limit to chroma

(note that after step 4 we normalize H to be between [0,1])

As a final step, instead of R we place H, instead of G we place S, and instead of B we place I.

Problem 5.1.2:

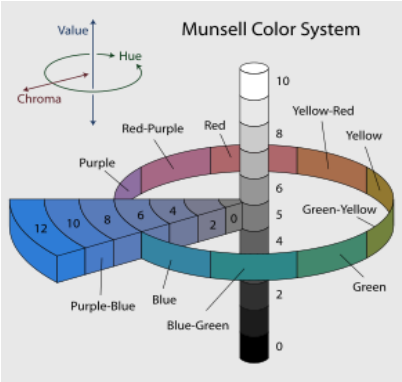
Present and discuss three new color spaces which were not introduced in class.

Solution:

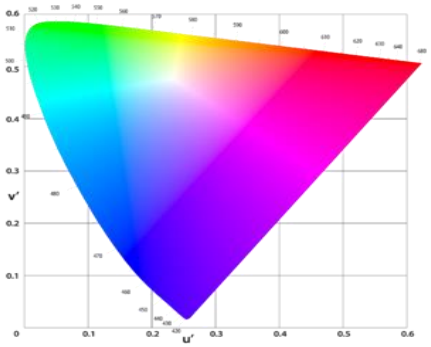
Munsell color system

the Munsell color system is a color space that specifies colors based on three properties of color: hue (basic color), chroma (color intensity), and value (lightness). It was created by Professor Albert H. Munsell In the first decade of the 20th century.

Several earlier color order systems had placed solid colors into a three-dimensional space of one form or another, but Munsell was the first to separate hue, value, and chroma into uniform and independent dimensions, and he was the first to illustrate the colors systematically in three-dimensional space.



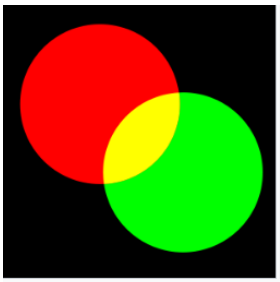
The Munsell color system, showing: a circle of hues at value 5 chroma 6; the neutral values from 0 to 10. The Chromas of purple blue are also of value 5.



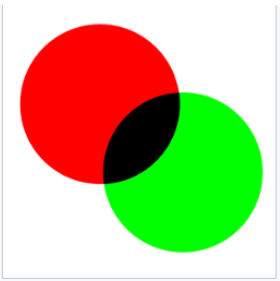
The (u', v') chromaticity diagram, also known as the CIE 1976 UCS (uniform chromaticity scale) diagram.

RG

The RG (also known as the red-green) color space is a color space that uses only two colors, red and green. It is an additive format, similar to the RGB color model but without a blue channel. Thus, blue is said to be completely out of the color gamut. This format is not in use today, and was only used on "two-color Technicolor" and other early color processes for films; by comparison with a full spectrum, its poor color reproduction made it undesirable. The system cannot create white naturally, and many colors are distorted.



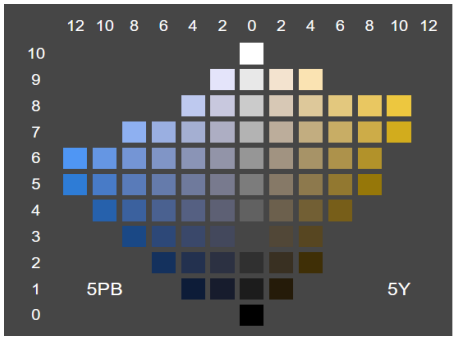
The additive RG color space can produce shades of black, red, green, and yellow.



The subtractive RG color space can produce shades of transparent (not white), red, green, and black.

As an example, here's an image of a parrot that is in the RG color space:

There is a similar color space called RGK which also has a black channel, so that images may become less distorted.



The Munsell value (vertical) and chroma (horizontal), when Hue is 5Y or 5PB

CIELUV

*The CIELUV color space (also referred to as the L*u*v* color space) is one of the uniform color spaces defined by the CIE in 1976.*

The CIE, in 1931, defined three standard primaries (X, Y, and Z) to replace red, green, and blue, because all visible colors could not be specified with positive values of red, green and blue components. the three results are called the tristimulus values. With this newly created X, Y, and Z primaries, all visible colors could be specified with only positive values of the primaries.

The values of L, u*, and v* are calculated according to the formulas below:*

$$L^* = \begin{cases} \left(\frac{29}{3}\right)^3 Y/Y_n, & Y/Y_n \le \left(\frac{6}{29}\right)^3 \\ 116(Y/Y_n)^{1/3} - 16, & Y/Y_n > \left(\frac{6}{29}\right)^3 \end{cases}$$

$$u^* = 13L^* \cdot (u' - u'_n)$$

$$v^* = 13L^* \cdot (v' - v'_n)$$

The quantities u'n and v'n are the (u', v') chromaticity coordinates of a "specified white object" – which may be termed the white point – and Yn is its luminance.

Equations for u' and v' are given below:

$$u' = \frac{4X}{X + 15Y + 3Z} = \frac{4x}{-2x + 12y + 3}$$

$$v' = \frac{9Y}{X + 15Y + 3Z} = \frac{9y}{-2x + 12y + 3}$$

Solution:

As stated in the previous section, each one of the RGB colors don't necessarily have to be quantized to the same level. To solve this problem we first extract the red and green layers and quantize them to 8 (which contains 3 bits) levels. Then we extract the blue layer and quantize it to 4 (which contains 2 bits) levels. Finally, We then put the colored channels together to form our new colored image.

Problem 5.2.3:

reduce the number of colors in an image to L levels with minimal visual quality loss.

Solution:

The method we described in problem 5.2.1 is a good way to achieve quantization, however there is a considerable amount of quality loss after applying it. There's also another way to achieve color quantization which makes use of the classic Unsupervised Learning algorithm known as K-means clustering. We consider our number of clusters the same amount as the possible values we want to achieve in the picture (K=L) for each of the pixels. And then for each pixel, we target the location of the closest centroid to then assign it to (Note that The algorithm assigns each centroid a color value that represents the average of all the pixels that were closest to that centroid). Finally, the result will be a higher quality quantization.

Note that the reason we say this algorithm outputs a higher quality image compared to the previous method is because, if for example we assign only 8 colors using k means (L=8), it Should be compared to when (L=2) in the previous formula. This is due to K-means having L possible values

Problem 5.2.1:

Implement uniform quantization on an image.

Solution:

In a previous article (Homework 1 to be exact) we explored the idea of performing quantization on a grayscale image. To recap, Quantization is the process of converting a continuous range of values into a finite range of discreet values. In the grayscale level, If we consider the image in which we want to do quantization on having grayscale values in range [0…255] (256 possible different values), for each pixel we can simply find:

v = floor((Image_matrix) ./ (256/L))

where L is the number of desired Levels we want to quantize by.

To perform quantization in the color space, we perform quantization separately for each pixel's Red, Green, Blue values (which each of them are also in a range of [0…255]) and then we finally put them alongside together to form a new colored pixel.

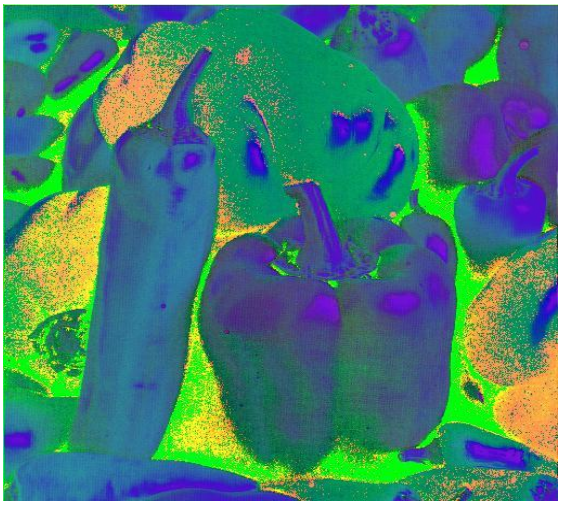
*Note that considering an N*N colored image, we have three separate N*N matrices, one for each of the three main colors. In turn, we perform quantization on each of the matrices and then we finally form a new image with the found values.*

Another thing to note is that we can perform quantization for all the colors on the same level, as well as different levels for different colors.

Problem 5.2.2:

For an image, quantize the R, G, and B components to 3, 3, and 2 bits,

If we put the HSI Components together, the resulting image will be as follows:



Which we can use to get useful information from the image. If we wanted we can lastly convert the image back to the RGB color space.

Problem 5.2.1:

We will perform quantization based on the formula we discussed previously, with various L's (which are all the same for red, green, and blue). Note that we get the MSE and PSNR between the output image and the original image for each of various L values:

~~~~ L = 8 ~~~~

```
MSE=295.1445 --- PSNR=23.4305
```

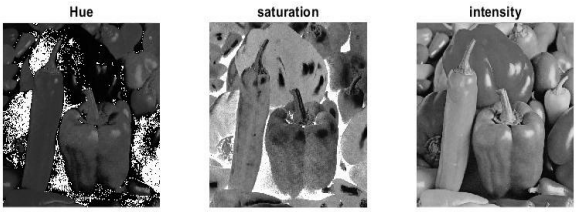
for each pixel, whereas the previous formula has L^3 possible values (one L for each color channel). If we put the output image's obtained by both methods side-by-side, we'll see the K-means giving a much better image, due to considering all the color values scattered through the original image, and takes averages of them to assign new values.

3- Describing the Results

Problem 5.1.1:

The original Pepper image is as follows:

If we display each of the Hue, Saturation, and Intensity as a separate image we get:



Which as can be seen most of the detial of the image is stored in the Intensity of an Image, while the color percieved in the image is related to Hue and Saturation.

~~~~ L = 64 ~~~~

MSE=3.3511 --- PSNR=42.879

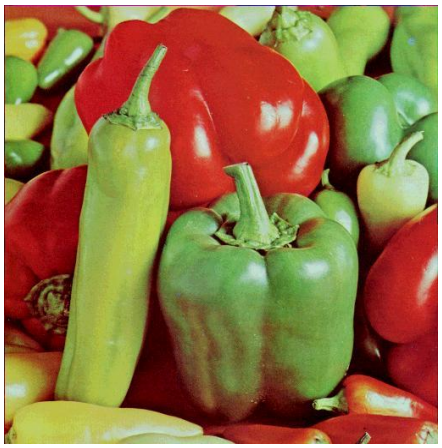
Again, we can't see too much difference when compared to when L=32, however if we compare this image to when L=16, we can see that a bit more detail is applied to the areas where light is reflected from the peppers.

Problem 5.2.2:

For the Pepper image If we quantize its red and green channels with 8 layers, and it's blue layer with 4 layers, the resulting image will be:

MSE=584.0113 --- PSNR=20.4666

~~~~ L = 16 ~~~~



MSE=73.3051 --- PSNR=29.4795

The colors here seem a bit more vibrant and colorful when we look at it and compare it to when L = 8, however the difference is hardly noticeable.

~~~~ L = 32 ~~~~

MSE=16.6718 --- PSNR=35.911

Hardly any difference can be seen with our eyes in this picture if we compare it to the previous one, other than it being a bit more colorful.
~~~~

~~~~ **L = 16** ~~~~

MSE=163.6068 --- PSNR=25.9928

In this output image that represents the original in 16 colors, we can see that more detail is visible

~~~~ **L = 32** ~~~~

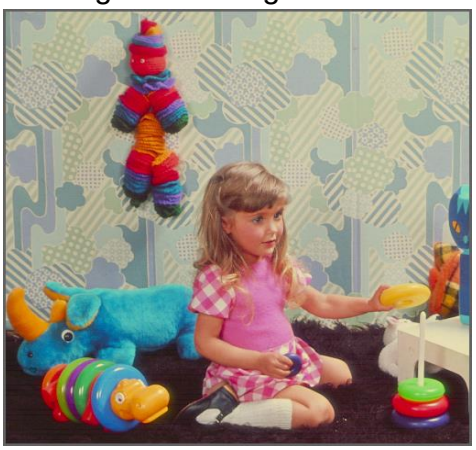
MSE=84.5704 --- PSNR=28.8586

And finally by taking L=32, we get an image that shows more detail, and is much more closer to the original image.

Which other than it having a lower quality than the original image, it still shows relatively vibrant colors, even though the blue channel of the original image is quantized to only 4 levels. This is due to the original image not containing too much blue, and it mostly being formed by it's red and green channels (which we consider 8 levels for each of these channels, which isn't a bad estimation to the original image's red and green channels).

Problem 5.2.3:

The original Girl image is as follows:



If we perform quantization using the k-means clustering algorithm, the result for different levels will be as follows:

~~~~ **L = 8** ~~~~

MSE=300.2498 --- PSNR=23.356

Using quantization we converted our image to be represented by only 8 colors here.
~~~~

Problem 5.2.2:

In this code, I apply the quantization formula for R and G considering 3 bits (8 layers), and for B considering 2 bits (4 layers)

```
img = imread('C:\Users\Arya\Desktop\computer vision\HW\Images\5\Pepper.bmp');
L1 = 8; %R and G layers
L2 = 4; % B layer
sz = size(img);

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

im(:,1) = R(:);
im(:,2) = G(:);
im(:,3) = B(:);
im = double(im);
%Doing the main computations for quantization
q1 = 256/L1;
v1 = floor(im(:,1:2)./q1); %R and G
v1 = v1.*q1;

q2 = 256/L2;
v2 = floor(im(:,3)./q2); %B
v2 = v2.*q2;

v = [v1 v2];
%reshape to the original image
out(:,:,1)=reshape(v(:,1),sz(1),sz(2));
out(:,:,2)=reshape(v(:,2),sz(1),sz(2));
out(:,:,3)=reshape(v(:,3),sz(1),sz(2));

out = uint8(out);
disp("MSE=" + immse(img,out) + " --- " + "PSNR=" + psnr(img,out));
imshow(out); imwrite(out,'3_quan_dif_levels.jpg');
```

Problem 5.2.3:

In the code below, I apply quantization by doing the K-means clustering algorithm.

```
clear;
img = imread('C:\Users\Arya\Desktop\computer vision\HW\Images\5\girl.bmp');
L = 8; %NUMBER OF LAYERS
sz = size(img);

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

im(:,1) = R(:);
im(:,2) = G(:);
im(:,3) = B(:);
im = double(im);
%Doing the main computations for quantization
[ind,centroids] = kmeans(im,L); %the k cluster centroid locations in the k-by-p matrix C
imQ = pdist2(im,centroids); %choosing the closest centroid to each pixel's R,G,B
[~,indMin]=min(imQ,[],2); %Target the cluster index of the closest distance.
v=centroids(indMin,:); %quantizing
%reshape to the original image
out(:,:,1)=reshape(v(:,1),sz(1),sz(2));
out(:,:,2)=reshape(v(:,2),sz(1),sz(2));
out(:,:,3)=reshape(v(:,3),sz(1),sz(2));

out = uint8(out);

disp("MSE=" + immse(img,out) + " --- " + "PSNR=" + psnr(img,out));
imshow(out);%imwrite(out,'4_quan_32.jpg');
```

4- Appendix(code)

Problem 5.1.1:

In the code below each of the HSI components are found separately, and finally in the end I put them all together in the output image (which is called HSI).

```
clear;
figure(1);
im = imread('C:\Users\Arya\Desktop\computer vision\HW\Images\5\Pepper.bmp');
img = double(im)/255; % show image between [0,1]

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

%FINDING HUE

theta = acosd((1/2*((R-G)+(R-B))) ./ ((((R-G).^2+((R-B).*(G-B))).^0.5)));
H = theta;
H(B>G)=360-H(B>G);
H=H/360; % normalize hue to between [0,1]

%FINDING SATURATION
S =1-(3./(R+G+B)).*min(img,[],3);
%FINDING INTENSITY
I = (R+G+B)./3;

HSI=zeros(size(im));
HSI(:,:,1)=H;
HSI(:,:,2)=S;
HSI(:,:,3)=I;
imshow(HSI);
```

```
figure(2);
subplot(1,3,1)
imshow(H);
title("Hue");
subplot(1,3,2)
imshow(S);
title("saturation");
subplot(1,3,3)
imshow(I);
title("intensity");
```

Problem 5.2.1:

In the code below, I apply the quantization formula for each of the R,G,B channels separately, and then finally place the channels in their correct format, to get the output image.

```
clear;
img = imread('C:\Users\Arya\Desktop\computer vision\HW\Images\5\Pepper.bmp');
L = 8; %NUMBER OF LAYERS
sz = size(img);

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

im(:,1) = R(:);
im(:,2) = G(:);
im(:,3) = B(:);
im = double(im);
%Doing the main computations for quantization
q = 256/L;
v = floor(im./q);
v = v.*q;
%reshape to the original image
out(:,:,1)=reshape(v(:,1),sz(1),sz(2));
out(:,:,2)=reshape(v(:,2),sz(1),sz(2));
out(:,:,3)=reshape(v(:,3),sz(1),sz(2));

out = uint8(out);
disp("MSE=" + immse(img,out) + " --- " + "PSNR=" + psnr(img,out));
imshow(out); imwrite(out,'2_quan2_8.jpg');
```