In The name of God

University Tehran

Engineering Facility

Electrical and Computer Engineering

Mechatronics

HW # 5

Amin Fadaeinedjad

810195442

Spring 98

Question 1

In this Question we are going to see that at fist what is color space and how can we change it for a certain data points.

A color space is a specific organization of colors. In combination with physical device profiling, it allows for reproducible representations of color, in both analog and digital representations. A color space may be arbitrary, with particular colors assigned to a set of physical color swatches and corresponding assigned color names or numbers such as with the Pantone collection, or structured mathematically as with the NCS System, Adobe RGB and RGB. A "color model" is an abstract mathematical model describing the way colors can be represented as tuples of numbers (e.g. triples in RGB or quadruples in CMYK); however, a color model with no associated mapping function to an absolute color space is a more or less arbitrary color system with no connection to any globally understood system of color interpretation. Adding a specific mapping function between a color model and a reference color space establishes within the reference color space a definite "footprint", known as a gamut, and for a given color model this defines a color space. For example, Adobe RGB and RGB are two different absolute color spaces, both based on the RGB color model. When defining a color space, the usual reference standard is the CIELAB or CIEXYZ color spaces, which were specifically designed to encompass all colors the average human can see.

At first we need to know that what is RGB, RGB stands for red green and blue which with the combination of all this colors we can make any color that we want, any color of this 3 is in the range of [0,255] and when we have all the color in the range 255 we will have the white light which means that is has all the base colors in it as well.

In the next part we must know that what is HSV color space in this case we know that is stands for Hue, Saturation and Value and we will talk about every one of this colors in this documentary.

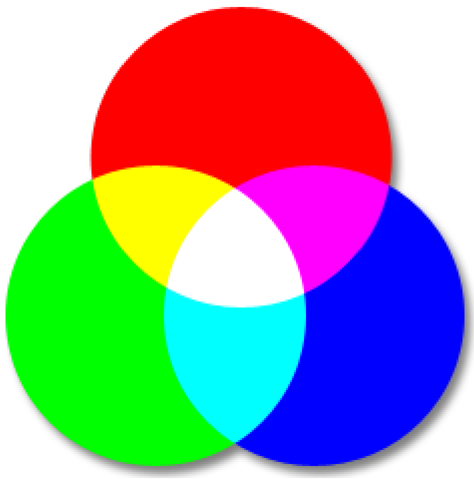

Figure 1

Figure 1 shows the combination of the three color in the color space

According to the following links that are mentioned in the references we know the mathematical formula that will give us the value for each parameter in the color space.

In these models, colors of each *hue* are arranged in a radial slice, around a central axis of neutral colors which ranges from black at the bottom to white at the top. The HSV representation models the way paints of different colors mix together, with the *saturation* dimension resembling various shades of brightly colored paint, and the *value* dimension resembling the mixture of those paints with varying amounts of black or white paint.

The simple answer is that unlike RGB, HSV separates *luma*, or the image intensity, from *chroma* or the color information. This is very useful in many applications. For example, if you want to do histogram equalization of a color image, you probably want to do that only on the intensity component, and leave the color components alone. Otherwise you will get very strange colors.

The color information is usually much noisier than the HSV information.

In computer vision you often want to separate color components from intensity for various reasons, such as robustness to lighting changes, or removing shadows.

Note, however, that HSV is one of many color spaces that separate color from intensity (See YCbCr, Lab, etc.). HSV is often used simply because the code for converting between RGB and HSV is widely available and can also be easily implemented. For example, the Image Processing Toolbox for MATLAB includes functions rgb2hsv and hsv2rgb.

RGB has to do with "implementation details" regarding the way RGB displays color, and HSV has to do with the "actual color" components. Another way to say this would be RGB is the way computers treats color, and HSV try to capture the components of the way we humans *perceive* color

At first we need to normalize the data from RGB so we will divide the data by 255 so the result will be between 0 and 1

$R' = \frac{R}{255} \quad G' = \frac{G}{255} \quad B' = \frac{B}{255}$

After that we will declare two other parameters

$$C_{\max} = \max(R', G', B') \qquad C_{\min} = \min(R', G'B')$$

$$\Delta = C_{\max} - C_{\min}$$

Know we are going to use this parameters to solve the value of the $H\ S\ and\ V$

$$if\ \Delta = 0 \rightarrow H = 0 \qquad else \rightarrow H = \begin{cases} 60 \times \left(\frac{G' - B'}{\Delta} \%6\right) & C_{\max} = R' \\ 60 \times \left(\frac{B' - R'}{\Delta} + 2\right) & C_{\max} = G' \\ 60 \times \left(\frac{R' - G'}{\Delta} + 4\right) & C_{\max} = B' \end{cases}$$

And the value of S will be

$$S = \begin{cases} 0 & , C_{\max} = 0 \\ \dfrac{\Delta}{C_{\max}} & , C_{\max} \neq 0 \end{cases}$$

And the value for V will be

$$V = C_{\max}$$

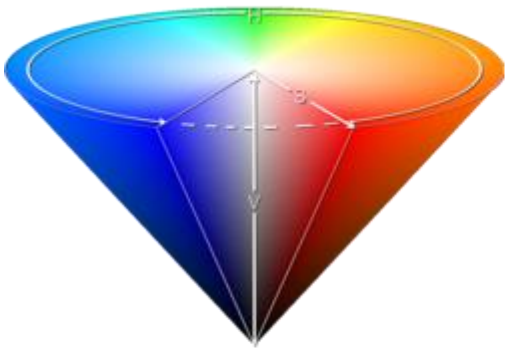

Figure 2

Figure 2 shows the color space for the HSV space and every parameter has its own meaning

Now here is a result for want we have done.

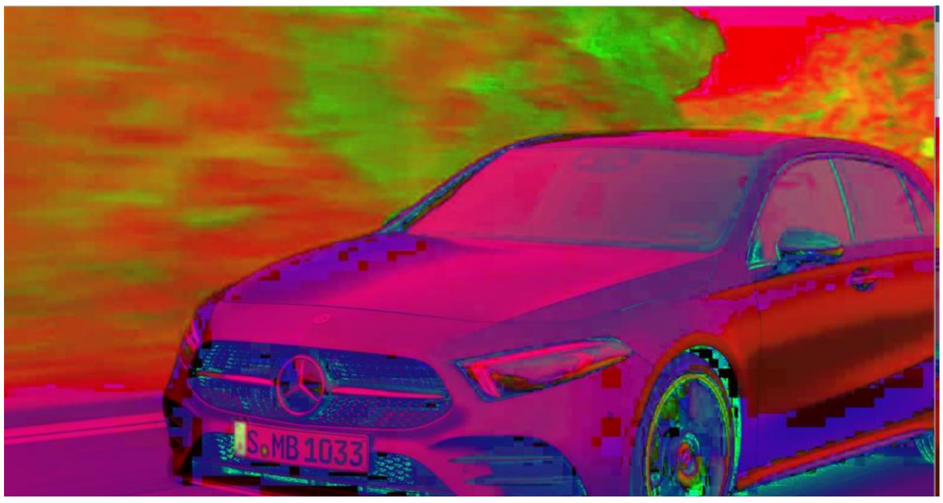

Figure 3

Figure 3 is the converted image from RGB to HSV using code that we have programed with the idea in the past page

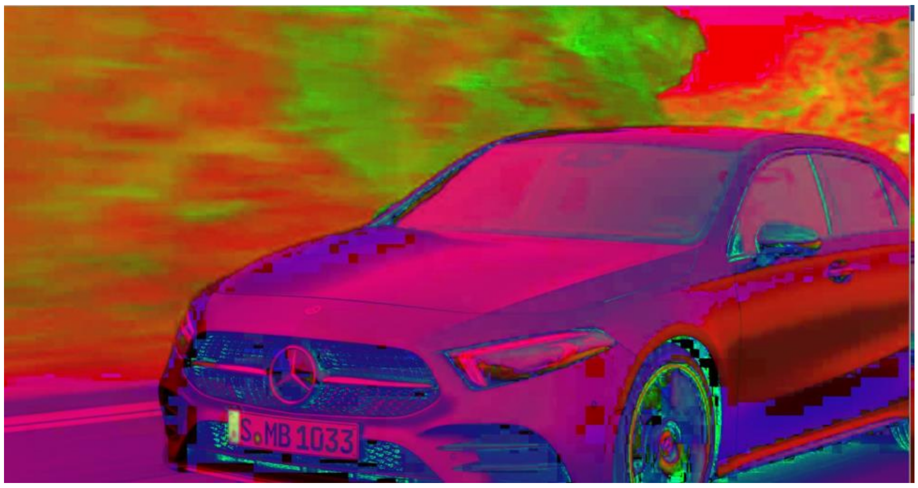

Figure 4

Image 4 is the result of the conversion using predefined code from openCV library

We used $NewImage = cv2.cvtcolor(img, cv2.COLOR\_BGR2HSV)$

And by comparing figure 3 and 4 we can see that we have done the conversion correctly and the original image was figure 5

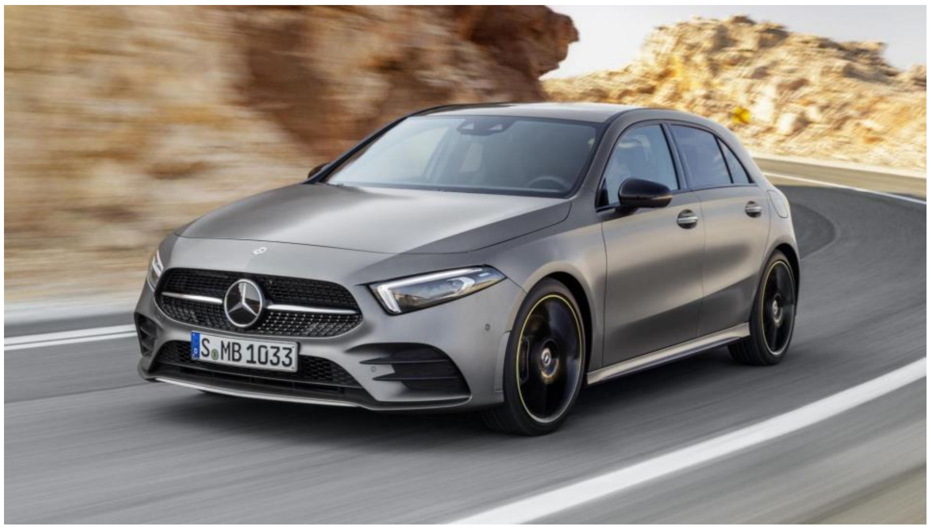

Figure 5

Because of the size of the window I wasn't able to make image 3 and 4 smaller.

We have made a lot of mistakes in the way to reach here where we are at first understanding the meaning of color space and the problem that we reached several times was that the cvt function in $OpenCv$ library wasn't the syntax of RGB but was BGR and that took a long time find out after the email that we received by the TA and one other thing is that the value of is between $0\ to\ 360$ and that is off limits to show in an image because the $imshow$ function does it in RGB colors so what I done was that in the end of the function I returned $\frac{h}{2}$ instead of h so we can see an image and one other bug that we had was that $imshow$ function cannot show fraction and even though that the our number had no fraction but still had .0 at the end and make us waste our time for such a thing ☹.

Question 2

In this question we have a data set with images of digits and we are going to recognize the other folder of images to see that can the contour see and find the image the advantages of this thing is that if we have a camera or any other thing that can take pictures of things we want to see that what digits and number they have.

So what we have done here was that we have a bunch of data and we want see can we classify them or now.

There is a folder that all the digits in it are standard so we will label is as the train data set and the other data points as the test data set

There are different methods of what we can do to recognize the data point, one of the things that we can use is the moments of the image and by comparing them with each other we can solve this question and one other method is to find the distance of the image and the train image form the folder and the minimum distance will show that label that the image will have.

Here are the results for every one of the two methods that we mentioned

Method #1: using the moments of the image

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$$

Figure 6

Figure 6 shows the definition of the moment where $I(x, y)$ is the color of the image which we need to turn it to grey image so we can use it as a binary image.

$$\begin{aligned}
h_0 &= \eta_{20} + \eta_{02} \\
h_1 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
h_2 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
h_3 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \qquad (6) \\
h_4 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (3\eta_{21} - \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\
h_5 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})] \\
h_6 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (
\end{aligned}$$

Figure 7

Figure 7 shows the $hu - moment$ that represents a number of the moments that can help us to classify the image.

So what we do at first we read the image and after that we need to convert the image to grey or to a black and white image, then we use the $hu - moment$

$$H_i = -\text{sign}(h_i) \log |h_i|$$

Figure 8

Figure 8 shows a way to normalize the $hu - moment$ so that if one moment was too big or small will not make bad results

And at the end we need to find the distance of the $hu - moments$ for every train data and the test data and the minimum distance will make the predicted label for it.

$$i^* = \min Norm_2(H_i, H_j) \rightarrow i^* \ is\ the\ label\ for\ image\ j$$

And so with this method we can predict the image the result is like this.

The predicted labels for scale is [0, 1, 0, 4, 0, 1, 7, 9, 7, 4]

And the accuracy is: 20.0 %

The predicted labels for rotation is [0, 1, 2, 3, 4, 0, 6, 7, 8, 9]

And the accuracy is: 90.0 %

The predicted labels for scale and rotation is [0, 1, 7, 7, 7, 5, 3, 7, 5, 5]

And the accuracy is: 40.0 %

We can see that still there are some miss classification and that could be because of the number of train data that are only 10 and that is too small.

Method #2:

In this part we are going to find the minimum distance of one image from the main image of the train data.

$$i^* = \min Norm\big(dictance\big(img(i), img(j)\big)\big) \rightarrow i^* \ is\ the\ label\ for\ image\ j$$

And with this method our result will be like this:

The predicted labels for scale is [0, 1, 7, 7, 7, 7, 7, 5, 5, 7]

And the accuracy is: 20.0 %

The predicted labels for scale is [0, 1, 2, 4, 2, 5, 6, 7, 8, 6]

And the accuracy is: 70.0 %

The predicted labels for scale is [0, 1, 5, 7, 5, 5, 1, 5, 5, 7]

And the accuracy is: 30.0 %

The method that we needed was quite a difficult thing to reach and we can still see that the accuracy is still low and we may use other things to make it work better like use the another $hu - momnet$ function with different scaling of different definition for distance.

And that was the algorithm that we used in this question.

Reference

https://dsp.stackexchange.com/questions/2687/why-do-we-use-the-hsv-colour-space-so-often-in-vision-and-image-processing

https://docs.opencv.org/trunk/df/d9d/tutorial_py_colorspaces.html

https://en.wikipedia.org/wiki/HSL_and_HSV