Universidad de Guadalajara

Centro Universitario de Ciencias Exactas e Ingenierías

Visión Robótica

Report 1- Converting RGB to HSV image

Luis Angel Muñoz Franco

Código: 303655636

Introduction:

Converting RGB to HSV

We start out by retrieving the red (R), green (G), blue (B) values, in a scale from 0 to 1, inclusively, as well as the largest and smallest of the R, G, B values, then calculate the difference between largest and smallest.

The "$scale_x$ " variable, below, represents the channel scale, e.g. 255.

$$R = \frac{R'}{scale_r}$$

$$G = \frac{G'}{scale_g}$$

$$B = \frac{B'}{scale_b}$$

$$m_{max} = max(R, G, B)$$

$$m_{min} = min(R, G, B)$$

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

Now, we get the hue, H, value. To do that, we look at the largest of the R, G, B, values. The smallest and second smallest element are subtracted off and divided by the difference between the largest and the smallest.

We then normalize the hue by adding either 0, 2, or 4. The resulting H is any real number. However, any arbitrary number below 0, and above 6 is considered redundant, and you may as well derive a value (H mod 6), or if H is negative, then (H mod 6) + 6, but it's not necessary, since a relatively decent HSV to RGB conversion algorithm should be able to work with any values of H.

$$H = \begin{cases} undefined, & \text{if } \Delta = 0 \\ \frac{G-B}{\Delta} & \text{if } m_{max} = R \\ \frac{B-R}{\Delta} + 2 & \text{if } m_{max} = G \\ \frac{R-G}{\Delta} + 4 & \text{if } m_{max} = B \end{cases}$$

$$H = H \ x \ scale_h$$

The brightness, V , is based on the brightest colour channel.

$$V = m_{max}$$

$$V = V \ \times scale_v$$

The saturation, S, is the difference between the largest and smallest colour channel values, divided by the brightness, V . If V is 0, then the resulting saturation is 0.

$$S = \begin{cases} 0 & \text{if } V = 0 \\ \frac{\Delta}{V} & \text{otherwise} \end{cases}$$

$$S' = S \ \times scale_s$$

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.

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`.

Code:

First I clean workspace:

```
clear
```

I read the image data to convert:

```
I = imread('/home/angel/Dropbox/135ls.jpg');
```

I take the size of the image and store the values in variables r, c and color:

```
[r,c,color]=size(I);
```

I storage the scale of RGB values in variable scale :

```
scale = 255;
```

I make a new variable to storage the new hsv image:

```
hsv = zeros(r,c,color);
```

The program runs through each row and column of the original image matrix data:

```
for i=1:r
    for j=1:c
```

I make a new variable 'rgb' to storage the 3 channels of colors normalized from 0 to 1, inclusively of the actual pixel in row 'i', column 'j':

```
        rgb = zeros(1,3);
        for k=1:3
            rgb(k) = double(I(i,j,k))/scale;
        end
```

I storage the largest value from the 3 channels of color and also I storage the index of that value.

```
        [mmax,imax]=max(rgb);
```

I storage the smallest value from the 3 channels of color and also I storage the index of that value.

```
        [mmin,imin]=min(rgb);
```

I calculate the difference between the largest and smallest value from the 3 channels of color in the actual pixel of the image and I storage it in variable 'def':

```
        dif= mmax-mmin;
```

I evaluate the conditions to calculate H as was described in the introduction:

```
        if imax == 1
            H = (rgb(2)-rgb(3))/dif;
        elseif imax == 2
            H = ((rgb(3)-rgb(1))/dif)+2;
        elseif imax == 3
            H = ((rgb(1)-rgb(2))/dif)+4;
        else
            H = nan;
        end
```

```
        if H < 0 || H > 6
            H = mod(H,6);
        end
```

I divide the Hue between the $scale_h = 6$:

```
        hsv(i,j,1) = H/6; %H
```

I storage the Value V multiplying the max value for $scale_v = 1$:

```
        hsv(i,j,3) = mmax*1; %V
```

I evaluate the conditions to calculate S:

```
        if ~hsv(i,j,3)
            hsv(i,j,2) = 0; %S
        else
            hsv(i,j,2) = dif/mmax*1;
        end
    end
end
```

I use Matlab function rgb2hsv to storage the image in HSV to compare with my code

```
hsvMatlab=rgb2hsv(I);
```

I evaluate if there is any difference between my HSV image and Matlab HSV image.

```
diferencia = [];
for i =1:r
    for j =1:c
        for k =1:3
            dd = round(hsv(i,j,k)-hsvMatlab(i,j,k), 10);
            if dd ~= 0 && ~isnan(dd)
                diferencia = [diferencia; i j k dd];
            end
        end
    end
end
```

I print differences, It shouldn't be any:

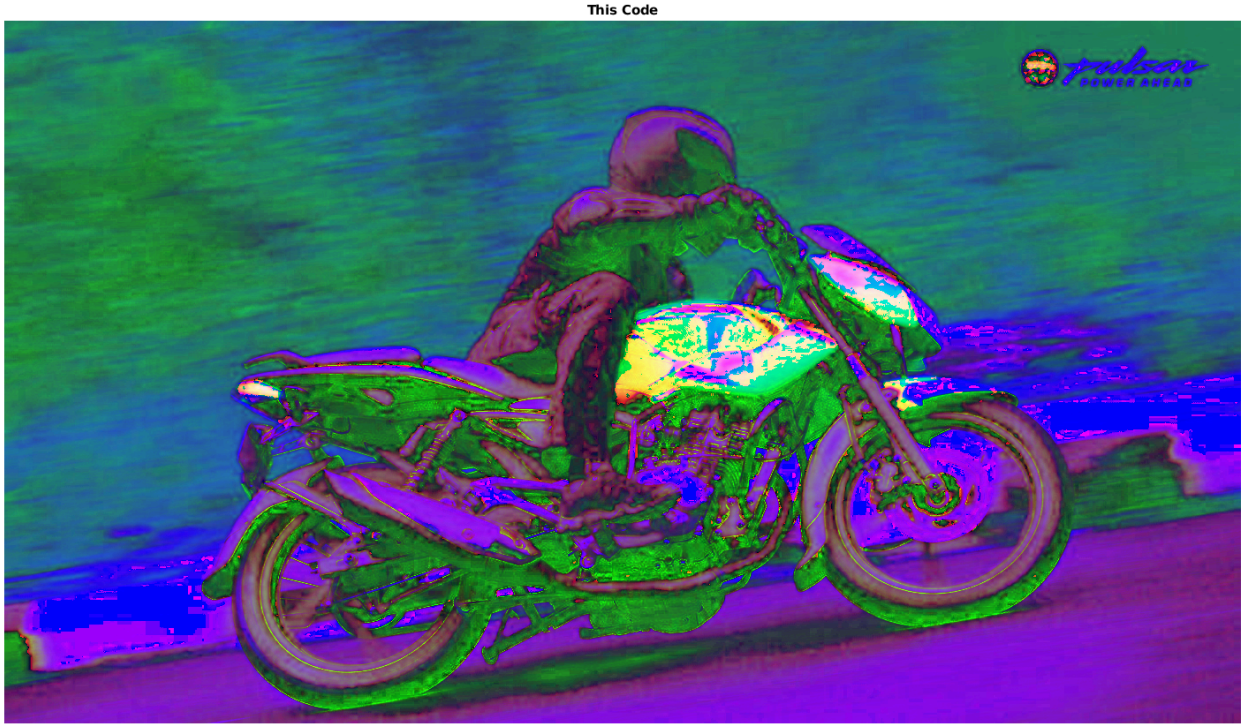
```
diferencia
```

```
diferencia =

     []
```

I print the image generated from this code and the image generated by Matlab function to compare both:

```
figure(1)
imshow(hsv)
title("This Code")
```




```
figure(2)
imshow(hsvMatlab)
title("Matlab rgb2hsv function")
```

Matlab rgb2hsv function