Lab 6 - Matlab And Color Images

Submitted by **Meldrick REIMMER, Danie SONIZARA, Awuraa Amma Okyere-BOATENG**

Contents

List of Figures

Chapter 1

Introduction

1.1 Background of Study

For our course module Sensors and Digitization we did a laboratory course with Matlab and color images. The main aim of this practical work was to study color images and some toolbox dedicated on image processing in Matlab and give a report about it.

Color is the phenomenon of human visual perception and the module of machine vision. Color information is widely used in the areas of virtual reality and human-computer interaction. Color is the product of a visual environment, illumination and the human brain. Research on color information representation and its processing is typically interdisciplinary. Color theory is based around the existence of three colors that, when mixed together, can produce all other colors. These colors, known as the primary colors, vary according to their application. Primary colors of light are red, green, and blue (RGB)

The RGB color model works exactly like those color receptors of the human eye work, the RGB color model describes a color by using 3 variables, Red, Green and Blue. These variables can be compared to the strength of the signals from the 3 types of color receptors in the nerves. A computer or TV screen works this way too,it has 3 types of cells, Red, Green and Blue, and can make each type brighter or darker independently, exciting the correct receptors of the eye to create the desired color. If you look with a magnifying glass to a white area of your computer screen, you can see that the color white is actually made out of the 3 colors red, green and blue. This means the white emitted by a computer screen is different from white sunlight, while white sunlight contains photons of all frequencies (except a few), the computer screen only has 3 frequencies. The human eye can't see the difference between these two kinds of white.

The RGB color model is also called the additive color model, because you add 3 color components together to form any color. In 24-bit color, each of the 3 components R, G and B is an 8-bit variable that can be an integer number between 0 and 255. 0 means the color component is off (black), while 255 means it's at its full intensity. 127 is half intensity. This means color 0,0,0 is the darkest black, color 255,0,0 is the brightest red, color 0,255,0 is the brightest green and color 0,0,255 is the brightest blue. 255,255,255 is the brightest white and 127,127,127 is gray. 32-bit color is the same but with an extra 8-bit alpha channel added that can be used for transparency of textures.

1.2 Objectives

During this practical work, we were given specific task to perform.Below are the tasks given:

- We to use the EO usb camera connected to the PC to take an image of color bricks.
- Export the image taken unto Matlab to perform other tasks.
- Using Matlab ,determine the size of this image and explain the format in which the size appears.
- Develop a program using Matlab to threshold in order to segment the red brick, yellow brick and blue brick each.
- Convert the image taken into HSV format.
- Study the effect of red lighting source on the threshold values.

These were the tasks we performed in this practical work.

Chapter 2

Hardware And Software Specifications

2.1 Hardware Specifications :

Two hardware components were used in this practical work.

1. PC Computer
2. LE (Lite Edition) EO USB 2.0

2.1.1 PC Component

A basic computer with a linux operating system was used with the needed Matlab and Ueye stream software installed on it.

Figure 2.1: Pc Computer

2.1.2 LE (Lite Edition) EO USB 2.0

The LE (Lite Edition) EO USB 2.0 cameras offer an even more compact, economical design in a high impact ABS plastic housing. Each camera features a 41mm x 41mm x 25.4mm housing and weighs only 32g.

Features :

- Ultra-Compact Housing
- Progressive Scan
- Adjustable Frame Rate via Binning, Sub-sampling or AOI
- Powerful Easy to Use Software Interface

Specifications :

- Manufacturer : EO
- Video Output : USB 2.0
- Mount : C-Mount

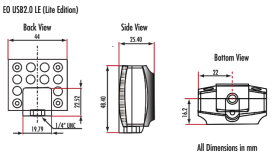

Figure 2.2: Camera Dimensions

Figure 2.3: LE(ite Edition) EO USB 2.0

2.2 Software Specifications :

Two types of software were used in this practical namely

1. Matlab software
2. Ueye stream software

2.2.1 Matlab Software

MATLAB is a high-performance language for technical computing. It integrates computation, visualization, and programming in an easy-to-use environment where problems and solutions are expressed in familiar mathematical notation. The name MATLAB stands for matrix laboratory. The matrix-based MATLAB language is the world?s most natural way to express computational mathematics. Built-in graphics make it easy to visualize and gain insights from data. The desktop environment invites experimentation, exploration, and discovery. These MATLAB tools and capabilities are all rigorously tested and designed to work together.

2.2.2 Ueye Stream Software

This was an interface for the camera to enable us take images and save them unto the PC Computer. We could also use it to change parameters The uEye Streaming library provides a simple mechanism to set up a RTSP server for streaming images captured with the camera. The uEye Streaming library supports automatic scaling and encoding of images with video codecs like MJPEG and H.264. A demo application is also included to set up a RTSP server and to stream images from a camera.

Chapter 3

Processing Color Image In RGB Format

3.1 Taking of the Image

We are to take an image of the color bricks. Before we did that, we had to set up the camera in the right position to be able to take a good image. These were the steps we took to take the image.

1. We run the software, which showed a live-view of the camera which was placed on a stand as seen in the figure below.

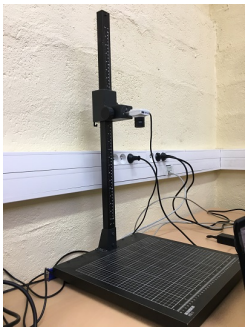

Figure 3.1: USB camera mounted on a stand

2. We placed the coloured bricks below the camera in order to be displayed on the live-view screen as shown below.

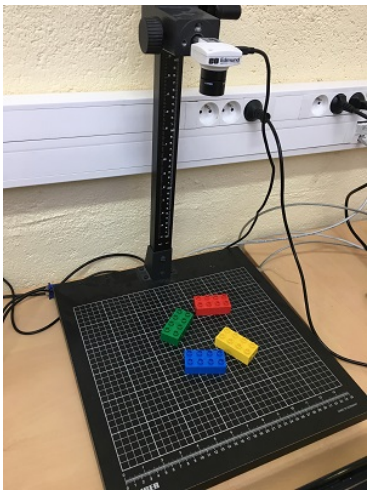

Figure 3.2: USB camera mounted with coloured bricks below

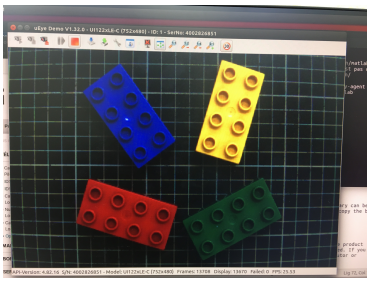

Figure 3.3: Live-view of the bricks on the Ueye software

3. The image appeared dark on the live-view. Therefore we selected settings menu on the top panel. In the 'Camera' settings we change the 'Gain' to auto because it was initially at 0.
4. We then saved the image taken of the bricks which was clearer and used that to perform the task asked in the next chapters.

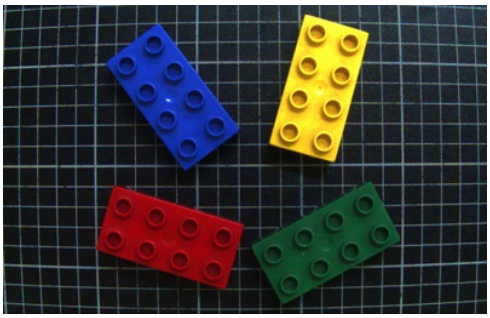

Figure 3.4: Image Chosen

3.2 Determining the Size of the Image

In this we determine the size of the image chosen by using Matlab. Below is the step in which we took to perform this task :

- After taking the image of the coloured bricks, we exported it into Matlab and determined its size using the following Matlab code :

```
1 -  close all;
2 -  clear all;
3 -  clc;
4
5    % Loading Image
6 -  bricks = imread('image.bmp');
7    % Getting the size
8 -  S = size(bricks);
```

Figure 3.5: Matlab Code

- This gives the result of the size in a matrix of the form (n;m;3) as shown below. This is a 3 dimensional matrix where n represents the length of the image, m represents the breadth and 3 specifies the colors in the image, which is Red, Green, Blue(RGB).
- Therefore the length of the image is 480, the breadth is 752 and it has 3 colours namely red, green and blue which are represented as RGB in Matlab.

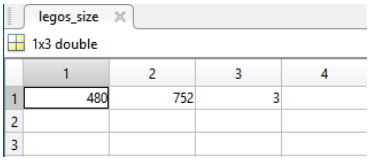

Figure 3.6: Showing size

3.3 Thresholding

In this part we had to create a program that creates a threshold in order to segment specific colors present in the image of the colored bricks taken.

Image thresholding is a simple, yet effective, way of partitioning an image into a foreground and background. This image analysis technique is a type of image segmentation that isolates objects by converting gray-scale images into binary images. Image thresholding is most effective in images with high levels of contrast.

Image segmentation is the process of partitioning an image into parts or regions. This division into parts is often based on the characteristics of the pixels in the image. For example, one way to find regions in an image is to look for abrupt discontinuities in pixel values, which typically indicate edges. These edges can define regions. Other methods divide the image into regions based on color values or texture.

Red Brick Segmentation

The first colored brick in the image we had to segment was the red brick. We wrote a code in Matlab to show only the red color which compose the image and other colors where all changed to gray-scale. Below is the code :

```
% Segmenting the red brick
redbricks = bricks
for r = 1:size(redbricks,1);
    for rn = 1:size(bricks,2);
        if redbricks(r,rn,1)< 80 || redbricks(r,rn,2) > 80 || redbricks(r,rn,3) > 100;

            % calculate greyscale
            gsc = 0.3 * redbricks(r,rn,1) + 0.59 * redbricks(r,rn,2) + 0.11 * redbricks(r,rn,3);
            redbricks(r,rn,:) = [gsc gsc gsc];
        end
    end
end

figure()
imshow(bricks)
title('Original Image')
figure()
imshow(redbricks)
title('Image with red brick segmented')
```

Figure 3.7: Matlab Code to Segment the red brick

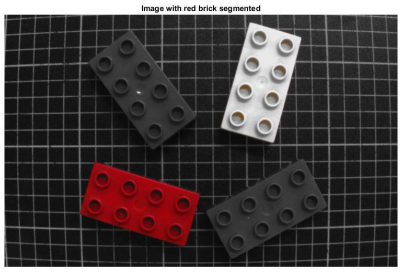

Figure 3.8: Resulted image of segmented red brick in Matlab

Yellow Brick Segmentation

The second colored brick in the image we had to segment was the yellow brick. We repeated a similar process to be able to segment the yellow brick for the other coloured bricks . The Matlab code therefore change to be as shown below :

```
% Segmenting the yellow brick
yellowbricks = bricks
for r = 1:size(yellowbricks,1);
    for rn = 1:size(yellowbricks,2);
        if yellowbricks(r,rn,1)< 80 || yellowbricks(r,rn,2) < 80 || yellowbricks(r,rn,3) > 100;

            % calculate greyscale
            gsc = 0.3 * yellowbricks(r,rn,1) + 0.59 * yellowbricks(r,rn,2) + 0.11 * yellowbricks(r,rn,3);
            yellowbricks(r,rn,:) = [gsc gsc gsc];
        end
    end
end
figure()
imshow(yellowbricks);
title('Image with yellow brick segmented')
```

Figure 3.9: Matlab Code to Segment the yellow brick

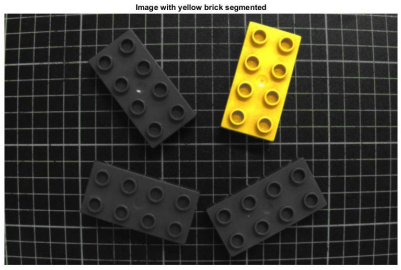

Figure 3.10: Resulted image of segmented yellow brick in Matlab

Blue Brick Segmentation

The last colored brick in the image we had to segment was the blue. we repeated the same process but with the aim of segmenting the blue brick from the rest of the colored bricks. Code Below:

```
% Segmenting the blue brick
bluebricks = bricks
for r = 1:size(bluebricks,1);
    for rn = 1:size(bluebricks,2);
        if bluebricks(r,rn,1) > 80 || bluebricks(r,rn,2) > 80 || bluebricks(r,rn,3) < 100;

            % calculate greyscale
            gsc = 0.3 * bluebricks(r,rn,1) + 0.59 * bluebricks(r,rn,2) + 0.11 * bluebricks(r,rn,3);
            bluebricks(r,rn,:) = [gsc gsc gsc];
        end
    end
end
figure()
imshow(bluebricks)
title('Image with blue brick segmented')
```

Figure 3.11: Matlab Code to Segment the blue brick

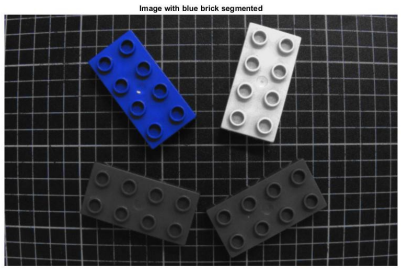

Figure 3.12: Resulted image of segmented blue brick in Matlab

Observation

We were able to segement each color of the bricks just asked. This was done with teh concept of RGB theory. We first looked at the values the images are having and having to know that we ten use it values to get the needed color we need. We just changed the other colors to gray scale leaving only the needed color in which we want to display. For the yellow brick we know its combination of green and red so we just made the parameters in such order for it to be shown.

Chapter 4

PROCESSING COLOUR IMAGE IN HSV FORMAT

4.1 HSV Format

HSV is an abbreviation for Hue, Saturation and Value and is an alternative method of representing RGB colour in computer graphics. In this model, the hue of each colour is arranged in a radial slice around an axis of neutral colours ranging from black at the bottom to white at the top. This representation shows how different colours are mixed together, the saturation resembles various shades of brightly coloured paint or colours and the value resembles the mixture of the various colours with varying amounts of black or white within them.

From the HSV cylinder shown below, it begins with red at 0° to green at 120°, blue at 240° and returning to red at 360°. The central axis ranges from black at a value of 0 to white at a value of 1.

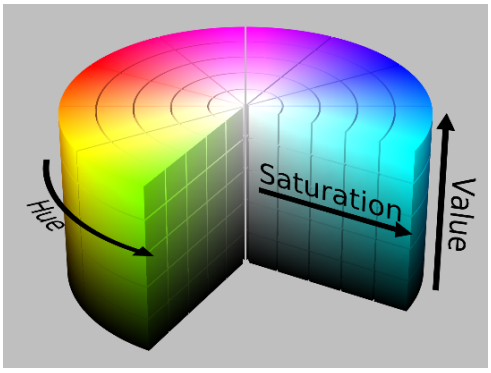

Figure 4.1: HSV Cylinder

4.2 Processing HSV in MATLAB

In this part we are to convert the image took of the bricks from RGB to HVS format. Before processing the HSV of the image of the coloured bricks, we had to convert the RGB image into an HSV image. We achieved this in Matlab using the following code :

```
65      %% Image in HSV format
66
67 -    bricks = imread('image.bmp');
68 -    brickshsv = rgb2hsv(bricks);
69 -    figure();
70 -    imshow(brickshsv);
```

Figure 4.2: Matlab code to convert RGB image into HSV

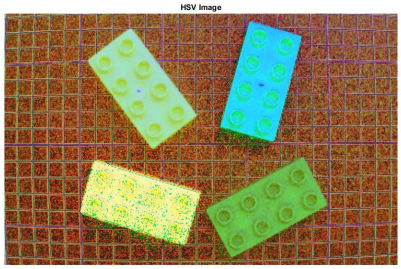

Figure 4.3: HSV Image

Afterwards, we obtained the hue, saturation and the value of the image using the Matlab code below :

```
% Defining HSV
H = brickshsv(:,:,1); % determines the hue
S = brickshsv(:,:,2); % determines the saturation
V = brickshsv(:,:,3); % determines the value

% Display
figure();
imshow(H);
figure();
imshow(S);
figure();
imshow(V);
```

Figure 4.4: Matlab code to convert RGB image into HSV

This gave us the result of the hue, saturation and value of the coloured HSV image as seen below.

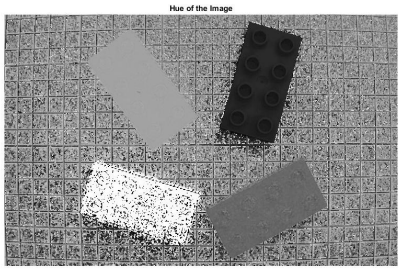

Figure 4.5: Hue-ratio of the Image

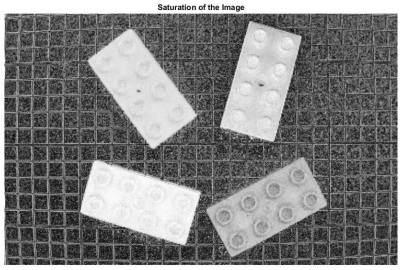

Figure 4.6: Saturation of the Image

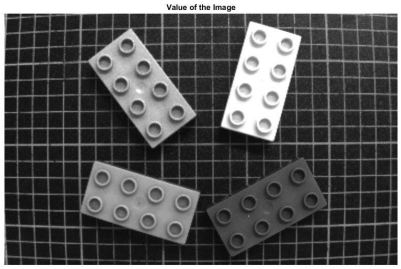

Figure 4.7: Value of the Image

Oberservation : The HVS provides a similar solution to change the RGB properties of the image. For example, consider that H represents R, S represents G, and V represents B. Regarding the coding, the rgb2hsv is syntax for changing systems, for example, some colormap is moved in arrangements of 0 and 1. With HVS there are input columns and output columns. The inputs represent what the level represents for RGB. The outputs show the hues and saturation.

Chapter 5

Lighting

In this part we study the effect of red light source on the threshold image in which we took in the previous part. Lighting has many effects on images. It is a hand in hand tool for image acquisition.

We first applied the red light source to the image and used the usb camera to snap a picture of it after which we will use to perform the task. We then converted the image into HSV in which we later segmented it into the Hue, saturation and Value.

5.1 Study of Red Lighting Source on an Image

Loading the image

```
87      %% Image with red ligthing  source
88      %load the image
89 -    brickRed= imread('imageR.bmp');
90 -    brickshsv = rgb2hsv(brickRed); % conversion to hsv format
91 -    figure();
92 -    imshow(brickshsv);
```

Figure 5.1: Code for loading image unto matlab

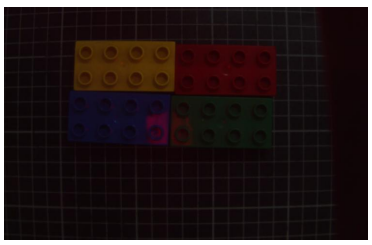

Figure 5.2: Red Light source image

Converting the Image Loaded into HSV and segmenting.

In this part we will convert the image in HSV and later segment it in order of preference. Below is the code and the results attained.

Results Below are the images in which we got for the red light source images.

```
84
85 -  %% Image with red ligthing  source
86 -  brickRed= imread('imageR.bmp');
87 -  brickshsv = rgb2hsv(brickRed);
88 -  figure();
89 -  imshow(brickshsv);
90
91 -  H = brickshsv(:,:,1);
92 -  S = brickshsv(:,:,2);
93 -  V = brickshsv(:,:,3);
94
95 -  figure();
96 -  imshow(H);
97 -  figure();
98 -  imshow(S);
99 -  figure();
   -  imshow(V);
```

Figure 5.3: code in matlab to perform task

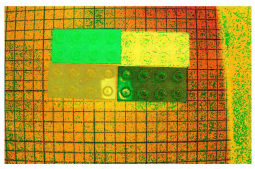

Figure 5.4: Image converted to HSV

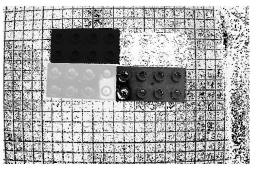

Figure 5.5: Hue

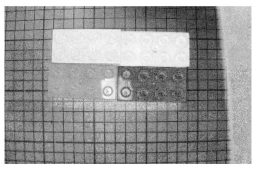

Figure 5.6: Saturation

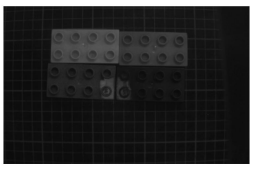

Figure 5.7: Value

Observation We can see from the images above that the red light source had much impact on the Value image, it made the picture more darker and the saturation image was became more high. The blue and green bricks with the red source had a different colors attained at both edges, it was clearly seen during the segmenting that it was able to remain noticed.

Chapter 6

Conclusion

We believe all the necessary tasks given was achieved and we gained more knowledge on the part of color imaging. We determined the size of the image we got and also performed segmentation on it. We also was able to change the image in to HSV and then also performed the study of red light source on an image.

Color is all around us, It is a sensation that adds excitement and emotion to our lives. Everything from the cloths we wear, to the pictures we paint revolves around color. Without color; the world (especially RGB World) would be a much less beautiful place.

This tutorial as given us more broader understanding on how colors can be manipulated in various ways and aspects. We will acquire more knowledge in this field and make deep research concerning this field.

Appendix

Matlab Code

```
% Authors: Danie SONIZARA - Meldrick REIMMER - Awuraa BOATENG
% MSCV1 - Sensors and Digitization
close all;
clear all;
clc;

% Loading Image
bricks = imread('image.bmp');
% Getting the size
S = size(bricks);

% Segmenting the red brick
redbricks = bricks
for r = 1:size(redbricks,1);
    for rn = 1:size(bricks,2);
        if redbricks(r,rn,1)< 80 || redbricks(r,rn,2) > 80 || redbricks(r,rn,3) >
    100;

            % calculate greyscale
            gsc = 0.3 * redbricks(r,rn,1) + 0.59 * redbricks(r,rn,2) + 0.11 *
    redbricks(r,rn,3);
            redbricks(r,rn,:) = [gsc gsc gsc];
        end
    end
end

figure()
imshow(bricks)
title('Original Image')
figure()
imshow(redbricks)
title('Image with red brick segmented')



% Segmenting the yellow brick
yellowbricks = bricks
for r = 1:size(yellowbricks,1);
    for rn = 1:size(yellowbricks,2);
        if yellowbricks(r,rn,1)< 80 || yellowbricks(r,rn,2) < 80 || yellowbricks(r,
    rn,3) > 100;

            % calculate greyscale
            gsc = 0.3 * yellowbricks(r,rn,1) + 0.59 * yellowbricks(r,rn,2) + 0.11 *
    yellowbricks(r,rn,3);
            yellowbricks(r,rn,:) = [gsc gsc gsc];
        end
    end
end
```

```
figure()
imshow(yellowbricks);
title('Image with yellow brick segmented')

% Segmenting the blue brick
bluebricks = bricks
for r = 1:size(bluebricks,1);
    for rn = 1:size(bluebricks,2);
        if bluebricks(r,rn,1) > 80 || bluebricks(r,rn,2) > 80 || bluebricks(r,rn,3)
    < 100;

            % calculate greyscale
            gsc = 0.3 * bluebricks(r,rn,1) + 0.59 * bluebricks(r,rn,2) + 0.11 *
    bluebricks(r,rn,3);
            bluebricks(r,rn,:) = [gsc gsc gsc];
        end
    end
end
figure()
imshow(bluebricks)
title('Image with blue brick segmented')


%% Image in HSV format
%load the image
bricks = imread('image.bmp');
brickshsv = rgb2hsv(bricks);        % conversion to hsv format
figure();
imshow(brickshsv);


% Defining HSV
H = brickshsv(:,:,1); % determines the hue
S = brickshsv(:,:,2); % determines the saturation
V = brickshsv(:,:,3); % determines the value

% Display
figure();
imshow(H);
figure();
imshow(S);
figure();
imshow(V);


%% Image with red ligthing  source
%load the image
brickRed= imread('imageR.bmp');
brickshsv = rgb2hsv(brickRed); % conversion to hsv format
figure();
imshow(brickshsv);

% Defining HSV for red light source image
H = brickshsv(:,:,1); % determines hue ratio
S = brickshsv(:,:,2); % determines the satutation
V = brickshsv(:,:,3); % determines the value

% Display
figure();
imshow(H);
figure();
imshow(S);
figure();
```

```
imshow(V);
```

bricksCode.m

References

1. **Light and Color**
   http://lodev.org/cgtutor/color.html#The_RGB_Color_Model_
2. **My Previous Lab Report in Bachelor**
3. **Matlab**
4. **HSV** https://en.wikipedia.org/wiki/HSL_and_HSV