Acquisition Practical

Labs 6 - Matlab and color images

Submitted by **Jaafer Al-Tuwayyij, Ivan Mikhailov and Selma BOUDISSA**

Contents

List of Figures

Introduction

A color image is a digital image that includes color information for each pixel. For visually acceptable result, it is necessary to provide three samples for each pixel xhich are interpreted as coordinates in some color space. The RGB color space is commonly used in computer displays, but other spacess such as YCbCR (color spaces used in video and digital photography.), HVS are often used in other contexts. A color image has three values or channels per pixel and theu measure intensity and chrominance of light. A color image is usually stored in memory as a raster map, a two-dimensional array of small integer triplets; or as three seperate raster maps, on for each channel.

Objectives

In this lab our objectives is to:

1. Study the color images.
2. Use matlab toolbox dedicated to image processing.
3. Be able to segment the color inside an image and display it in different figure.
4. Convert the image into HSV format.
5. Study the effect of a red lighting source on the image.

Chapter 1

Hardware specifications

1.1 Matlab software

Definition: Matlab mean matrix laboratory is a multi-paradigm numerical computing environment. Matlab allow matrix manipulations, plotting of functions and data implementation of algorithms, creation of user interfaces...

Image processing with Matlab: Image Processing Toolbox provides a comprehensive set of reference-standard algorithms and workflow apps for image processing, analysis, visualization and algorithm development. You can perfom image segmentation, image enhancement, noise reduction, geometric transformation, image registration and 3D image processing.

Advantages:

- A very large database of built-in algorithls for image processing and computer vision applications.
- Ability to call external libraries such as OpenCV
- Process both images and video
- Large user community with lots of free code and knowledge sharing
- ...

1.2 Camera EO Edmund

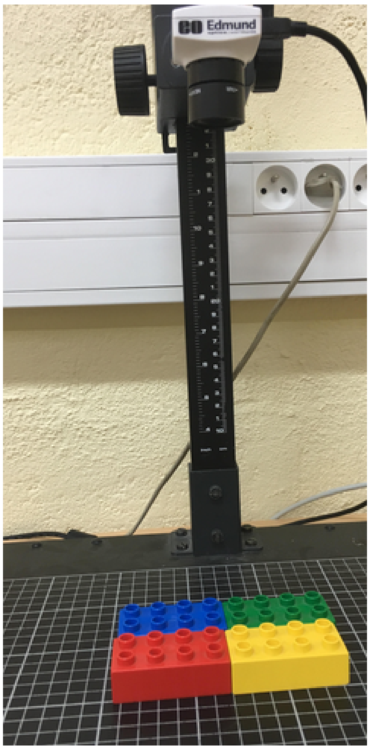

Figure 1.1: EO Edmud camera

1.2.1 Features

- Ultra-Compact Housing
- Progressive Scan
- Adjustable Frame Rate via Binning, Sub-sampling or AOI
- Powerful Easy to Use Software Interface
- Video output USB 2.0
- CMOS system

1.2.2 Description

The EO USB 2.0 camera is equipped with a host of features to make machine vision easy. Included with each camera is software interface that allows users to set a specified Area of interest (AOI), gain, exposure time, frame rate, trigger delay and select digital output(flash) delay and duration. The camera software allows capturing of still images in JPEG and Bitmap file format.

Chapter 2

Colour images in RGB format

2.1 Original image

The first step of this lab is to take a picture using the camera EO Edmud that we will use during the lab experiments.

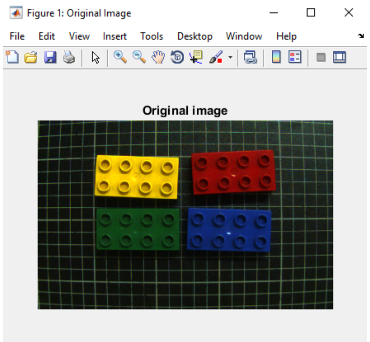

Figure 2.1: Original image of the bricks

2.2 Size of the image

```
image = imread('colorbricks.bmp');          % Load the orginal image
s = size(image);                            % Determine the size of the image
```

size.m

Explanation: We determined the size of the image by using the matlab function called size. Once I compile the code above I obtain the result of the size of the original image taken using the camera EO Edmud. The size of this image is a (n;m;3) matrix which correspond to the number of rows (n) by the number of columns (m) and 3 correspond to the number of colors which is RGB.

2.3 Basic program to threshold the image

The purpose of this part is to develop a basic program to threshold the image in order to segment the red, the yellow and also the blue bricks.

2.3.1 Segmentation of the Red brick

```
redbrick=image;
for mm = 1:size(redbrick,1)
    for nn = 1:size(redbrick,2)
        if redbrick(mm,nn,1)<80 || redbrick(mm,nn,2)>80 || redbrick(mm,nn,3)>100;
 gsc = 0.3*redbrick(mm,nn,1)+0.59*redbrick(mm,nn,2)+0.11*redbrick(mm,nn,3);
 redbrick(mm,nn,:) = [gsc gsc gsc];
        end
    end
end
```

basicthresholdredbrick.m

Explanation: First of all this picture as many rows and many columns and we are going to make a comparaison between each rows and columns and compare it to a certain ratio and to RGB. So we are going to change it to gray scale in the first time. So nn in the code above is for the rows and mm is for the columns. We change it to gray scale by using the code in the 4th line (above code).

1. if redbrick(mm,nn,1)<80
   We choose the number 80 to control if the red columns is less than the value of 80. It can be any number between 0 and 255.

2. || redbrick(mm,nn,2)>80
   If the green color is greater than 80.

3. || redbrick(mm,nn,3)>100;
   if the blue color was larger than 100.

So we are going to see if the pixel of that picture is involve within any of this category. And then we calculate the gray scale 'gsc' ligne 5, formula extract from the matlab function gray scale. 30% of the red color and 59% of the green color and 11% of the blue color.

Result: This is the result obtain using the basic program to threshold in order to segment the red brick.

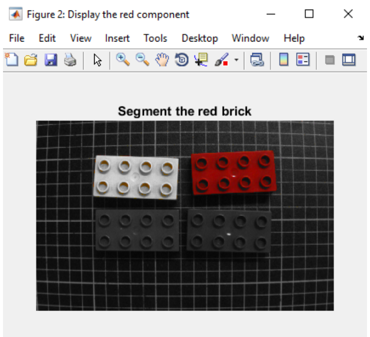

Figure 2.2: Segmentation of the red brick after thresholding

2.3.2 Segmentation of the Yellow brick

```
if yellowbrick(mm,nn,1)<80 || yellowbrick(mm,nn,2)<80 || yellowbrick(mm,nn,3)>100;
```

basicthresholdyellowbrick.m

Result This is the result obtain using the basic program to threshold in order to segment the yellow brick

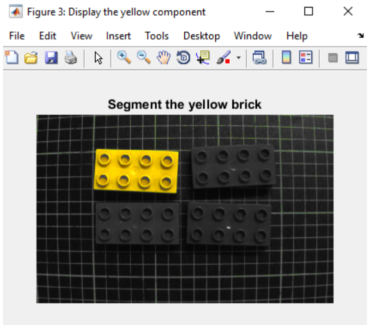

Figure 2.3: Segmentation of the yellow brick after thresholding

2.3.3 Segmentation of the Blue brick

```
if bluebrick(mm,nn,1)>80 || bluebrick(mm,nn,2)>80 || bluebrick(mm,nn,3)<100;
```

basicthresholdbluebrick.m

Result This is the result obtain using the basic program to threshold in order to segment the red brick

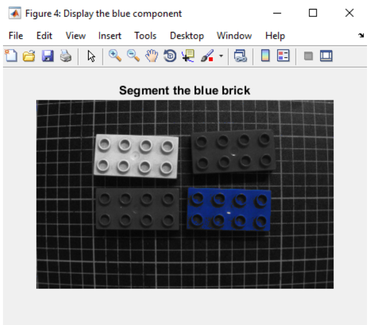

Figure 2.4: Segmentation of the blue brick after thresholding

Chapter 3

Colour images in HSV format

3.1 HSV format

```
hsvimage = rgb2hsv(image);  % conversion  to HVS format
```

hsvformat.m

Result This is the result obtain by converting the original image into the hsv format using the matlab function 'rgb2hsv'.

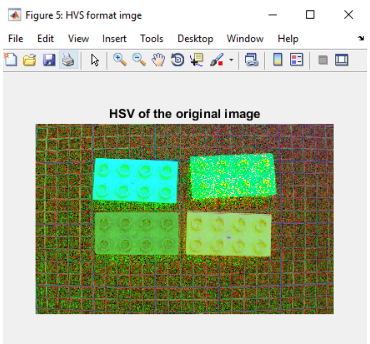

Figure 3.1: Conversion of the original image to HSV format

3.2 Basic segmentation

```
% Definition of the HSV
H=hsvimage(:,:,1); % Display the hue ratio
S=hsvimage(:,:,2); % Display the Staturation
V=hsvimage(:,:,3); % Display the value
```

hsv.m

Result This is the result obtain by defining the H, S and the V of the the image.

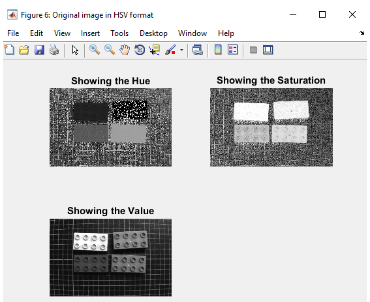

Figure 3.2: Basic segmentation using HSV format of the image

Chapter 4

Lighting

The target of this part of the lab is to apply a red light source on the bricks and capture that image using the same camera as previously.

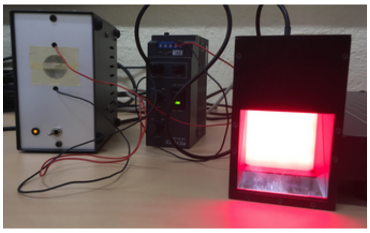

Figure 4.1: Red light

4.1 Applying red lighting source

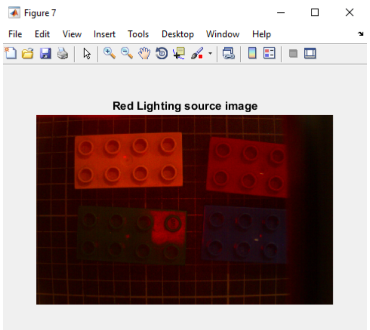

Figure 4.2: Image obtain applying a red lighting source

4.2 Threshold values obtained

Based on the previous part we just define the H, S and V in this image with the red light applied.

```
% Conversion to HSV format
redsourcehsv =rgb2hsv(redsource);
figure()
imshow(redsourcehsv)
title('Red source lighting image in HSV format');
% Definition of the HSV
H2=redsourcehsv(:,:,1); % Display the hue ratio
S2=redsourcehsv(:,:,2); % Display the Staturation
V2=redsourcehsv(:,:,3); % Display the value
```

hsvredlight.m

Result This is the result obtain by defining the H, S and the V of the the red lighting source image.

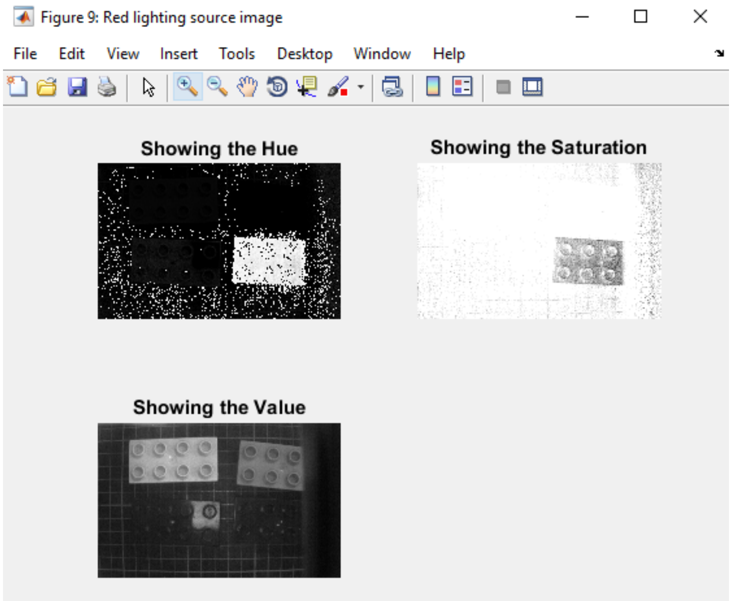

Figure 4.3: Basic segmentation using HSV format of the red light source image

4.3 Comparaison

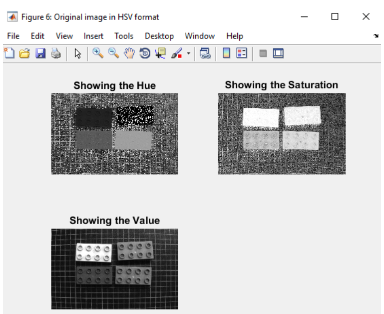

Figure 4.4: HSV definition for orignal image

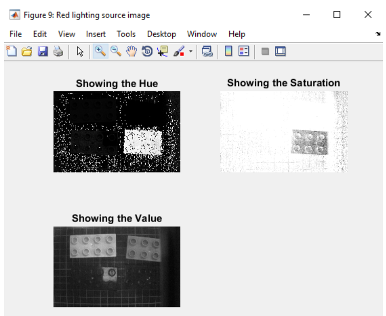

Figure 4.5: HSV definition for red lighting image

Explanation: As you can see the red light source have an effect on different point. When we observe the Hue ratio image for example during the application of the red light it has complitely turn the red brick (which is in the top right corner of the original image) into black, same for the yellow brick. But we can see that the blue brick(which is in the bottom right corner of the original image) turn into white color. The saturation of the image applying red lighting source turn the all image into white we can only see the shape of the blue brick. For the value of the image we can only see the yellow brick and red brick shapes. Finally we can conclude that the red lighting as a effect on the threshold values of the image.

Chapter 5

Conclusion

In this lab we were able to complete all the objectives. We also realize that Matlab is one of the best software for Image Processing and traitement of color image which also provide great implemented tools for the rgb and many other functions. We were able to segment each bricks color, convert the image into HSV format and study the effect of the red lighting source on a RGB image.

Appendix

- **Matlab source code**

```
%% Lab 6 - Matlab and color images
% Date: 28/11/17
% Group 1A: Ivan, Jaafer and Selma

close all

%% Colour images in RGB format
image = imread('colorbricks.bmp');          % Load the orginal image
figure('Name', 'Original Image')            % Open a new figure
imshow(image);
title('Original image');
s = size(image);                            % Determine the size of the image
%------------------------------COMMENT------------------------------------
% The size of the matrix is a (490; 752; 3).
% It's a (n;m;3) matrix cause it is a row by column matric with 3 which
% correspond the number of color RGB.
%-------------------------------------------------------------------------

% Basic programm to threshold the image
% show the red brick
redbrick=image;
for mm = 1:size(redbrick,1)
    for nn = 1:size(redbrick,2)
        if redbrick(mm,nn,1)<80 || redbrick(mm,nn,2)>80 || redbrick(mm,nn,3)
    >100;
 gsc = 0.3*redbrick(mm,nn,1)+0.59*redbrick(mm,nn,2)+0.11*redbrick(mm,nn,3);
 redbrick(mm,nn,:) = [gsc gsc gsc];
        end
    end
end
figure('Name', 'Display the red component')
imshow(redbrick);
title('Segment the red brick');


% show the yellow brick
yellowbrick=image;
for mm = 1:size(yellowbrick,1)
    for nn = 1:size(yellowbrick,2)
        if yellowbrick(mm,nn,1)<80 || yellowbrick(mm,nn,2)<80 || yellowbrick(mm
    ,nn,3)>100;
 gsc = 0.3*yellowbrick(mm,nn,1)+0.59*yellowbrick(mm,nn,2)+0.11*yellowbrick(mm,
    nn,3);
yellowbrick(mm,nn,:) = [gsc gsc gsc];
        end
    end
end
figure('Name', 'Display the yellow component')
imshow(yellowbrick);
title('Segment the yellow brick');


%show the  blue brick
bluebrick=image;
for mm = 1:size(bluebrick,1)
    for nn = 1:size(bluebrick,2)
        if bluebrick(mm,nn,1)>80 || bluebrick(mm,nn,2)>80 || bluebrick(mm,nn,3)
    <100;
 gsc = 0.3*bluebrick(mm,nn,1)+0.59*bluebrick(mm,nn,2)+0.11*bluebrick(mm,nn,3);
 bluebrick(mm,nn,:) = [gsc gsc gsc];
```

```
        end
    end
end
figure('Name', 'Display the blue component')
imshow(bluebrick);
title('Segment the blue brick');


%% Colour images in HVS format

hsvimage = rgb2hsv(image);  % conversion  to HVS format
% Displaying the figure
figure('Name', 'HVS format imge')
imshow(hsvimage);
title('HSV of the original image');

% Definition of the HSV
H=hsvimage(:,:,1); % Display the hue ratio
S=hsvimage(:,:,2); % Display the Staturation
V=hsvimage(:,:,3); % Display the value
figure('Name', 'Original image in HSV format')
subplot(2,2,1), imshow(H), title('Showing the Hue')
subplot(2,2,2), imshow(S), title('Showing the Saturation')
subplot(2,2,3), imshow(V), title('Showing the Value')


%% Lighting

% Load the lighting red source image

redsource = imread('redsource.bmp')
figure()
imshow(redsource);
title('Red Lighting source image');

% Conversion to HSV format
redsourcehsv =rgb2hsv(redsource);
figure()
imshow(redsourcehsv)
title('Red source lighting image in HSV format');
% Definition of the HSV
H2=redsourcehsv(:,:,1); % Display the hue ratio
S2=redsourcehsv(:,:,2); % Display the Staturation
V2=redsourcehsv(:,:,3); % Display the value
figure('Name', 'Red lighting source image')
subplot(2,2,1), imshow(H2), title('Showing the Hue')
subplot(2,2,2), imshow(S2), title('Showing the Saturation')
subplot(2,2,3), imshow(V2), title('Showing the Value')
```

lab6.m

References

- **Definition of color images** https://en.wikipedia.org/wiki/Color_image
- **Definition of Matlab** https://en.wikipedia.org/wiki/MATLAB
- **Camera EO Edmud specifications** https://www.edmundoptics.com/cameras/usb-cameras/EO-USB-2.0-CMOS-Machine-Vision-Cameras/
- **Youtube video about RGB** https://www.youtube.com/watch?v=BmwFEcByLdY