COMPUTER VISION LABORATORY

SESSION 4: Color-based segmentation

June 17, 2020

Arlotta Andrea 4089306
Cantoni Francesca 4698289

Contents

Chapter 1

Abstract

This laboratory experience aims to analyze the HSV and the RGB characteristics of a set of images in order to detect a specific region. Specifically it was required to:

1. compare the visual characteristics of each photo, by displaying their RGB, grayscale and HSV channels
2. note the variation of the RGB components and of the Hue one in the area of the red car on the street, with the auxiliary of a cropped image [1]
3. estimate the mean and standard deviation value of the Hue parameter for the body of the red car
4. detect and segment the red car by thresholding the Hue component

Description of the paragraphs:

Inside **'Introduction'** it is given to the reader a brief explanation about the necessity of image-segmentation for colors and/or objects detection.

The next section **'Theoretical information'** provides information about human vision system and color-based image segmentation.

Chapter **'Matlab'** shows the dependencies between each function through a tree function and inside **'Conclusion'** chapter are presented and commented all our outcome results.

[1] The user can also detect and segment the red car with a RGB blob detection by uncommenting the auxiliary section of the code.

Chapter 2

Introduction

The visual perception of the world provides information about the environment that surround us: in general humans can easily manage all these information without being overburdened.

Conversely computers, given as input an image, can barely understand what/who is the subject of the photo. Mathematical tools make possible to cut off unnecessary information from the image and to highlight the important one in such a way to extract the major real world properties of the subject.

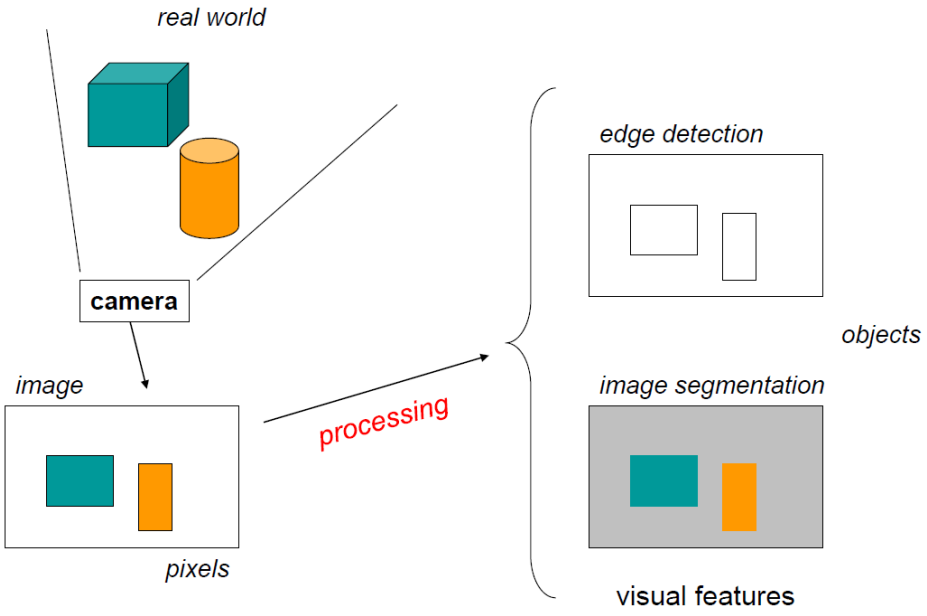

Figure 2.1: Main steps for obtain visual features starting from the real world

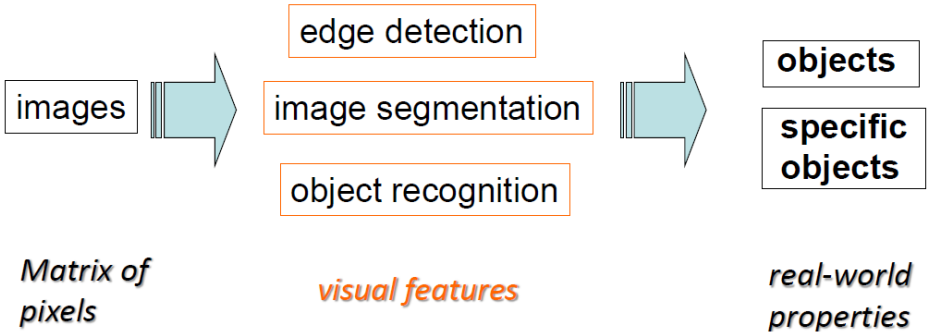

Figure 2.2: More specific way to represents computer image processing

The process that leads to the partition of a digital image into multiple regions is defined as **segmentation of the image**[1].

Segmentation aims is to divide an image into regions, each one composed by pixels that share the same characteristics, in order to extract areas that may be more representative and easier to analyze.

Inside this laboratory session we have performed two different types of segmentation each one that implement a different type of color-based mask.

[1]A more detailed description of a specific type image segmentation, the color-based method, will be provided inside section **3.3 Image segmentation based on color.**

Chapter 3

Theoretical information

Before colors and color-based segmentation topics it is important to make some recall about image formation.

3.1 Image formation

As mentioned during the previous reports, the image formation system is composed by these three main blocks:

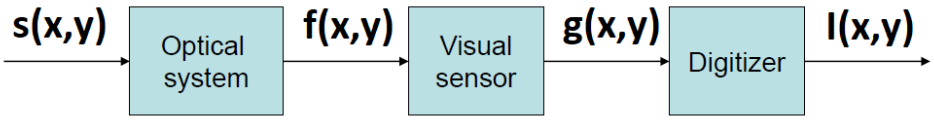

Figure 3.1: Image formation system

- **Optical system:** the input-output of this block is described by a 2D convolution between **s(x,y)** and the point spread function **h(x,y)**.
  In particular **s(x,y)**, also called intensity or gray level signal, is the result of the product between the illumination **il(x,y)** and the reflectance **r(x,y)** at the same point.

$$s(x,y) = il(x,y) * r(x,y) \quad \begin{cases} 0 < il(x,y) < \inf \\ 0 < r(x,y) < 1 \end{cases} \tag{3.1}$$

  This behaviour can be seen in the following figure.

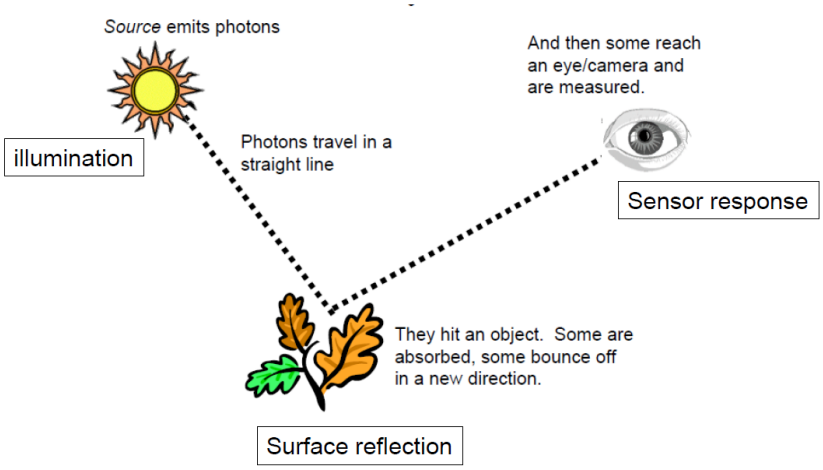

Figure 3.2: Photometry overview

- **Visual sensor:** sensor that transduces the resultant intensity signal into an electric one (i.e camera)
- **Digitizer:** converts the 2D analog signal into a digital one through a A/D convert. This operation is performed using the sampling [1] and the quantization[2] technique. The resultant image **I(x,y)** is affect by errors due to the digitalization step and noise contribution.

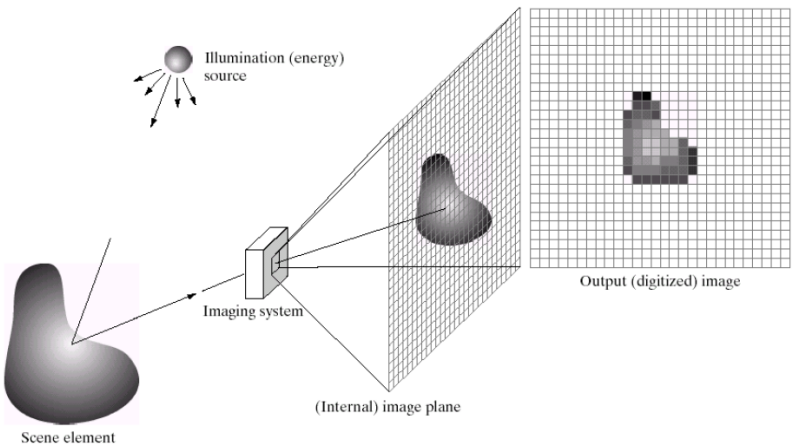

Figure 3.3: Main steps of digital image formation

[1] Sampling the 2D space on a regular grid.

[2] Quantizing the amplitude of each sampled value rounding it to the nearest integer. Grayscale images are quantized with 256 levels, thus they required 1 byte for the representation of each pixel, instead of binary ones that required only two levels [0,1].

3.2 Color fundamentals

In order to perform color-based detection is important first to understand how human perceives different colors.
Human vision system responds differently to different wavelengths λ [nm] of light accordingly to these figures.

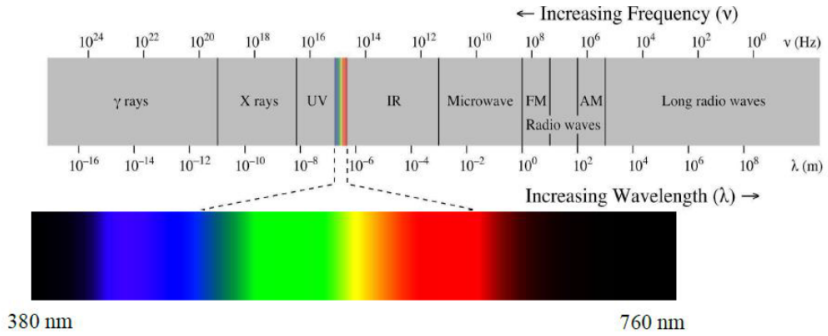

Figure 3.4: Full spectrum of wavelengths

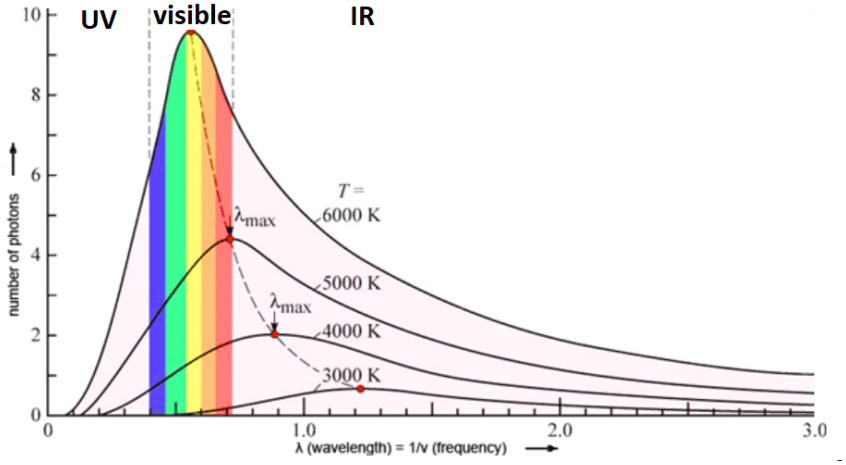

Figure 3.5: Zoom of the behaviour of the visible wavelengths

Using this information it is possible to redefine the concept of image perception considering the human eye as the visual sensor[3] so, recalling the previous notation, $\mathbf{il} = illumination$, $\mathbf{r} = reflectance$ and $\mathbf{s} = intensity$, the new formalization can be described by **Figure 3.6**.

[3]It is no longer necessary to perform the digitization because now information are processed through the brain.

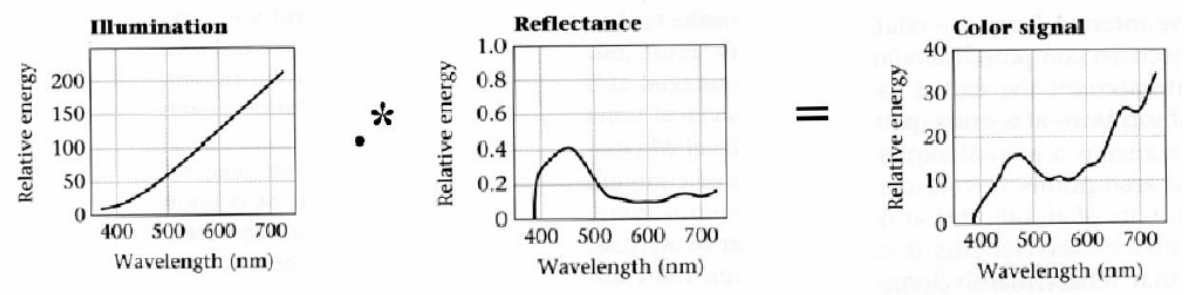

Figure 3.6: Interaction of light and surfaces

Another important concept is the **trichromacy principle**, discovered through the *color matching experiment*, that defines that all visible colors can be obtained as a weighted sum of three colors the **RGB colors** (R=red, G=green, B=blue).

This phenomenon can be explained by assuming that there are three distinct types of color transducer in human eye, as it possible to see from the figure below.

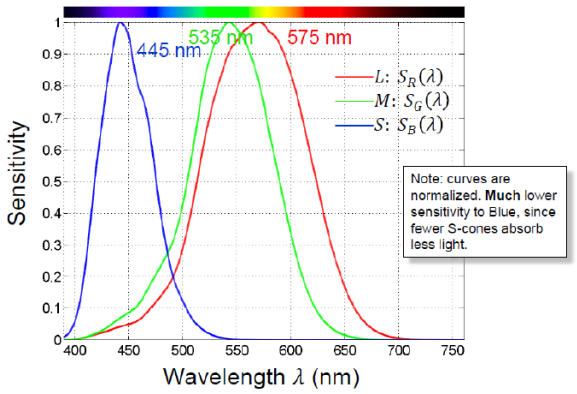

Figure 3.7: Absorption of light in the cones of the human retina

The three cones act as filters on the image spectrum thus to get the output of each filter is necessary to multiply its response curve, S_R), S_G) and S_B), to the input spectrum $\mathbf{C}(\lambda)$ and integrate the result over all wavelengths in such a way to obtain each effective cone stimulation, respectively α_R, α_G and α_B.

3.2.1 Color spaces

Color spaces are invented to simplify color management within different contexts.
The main models are:

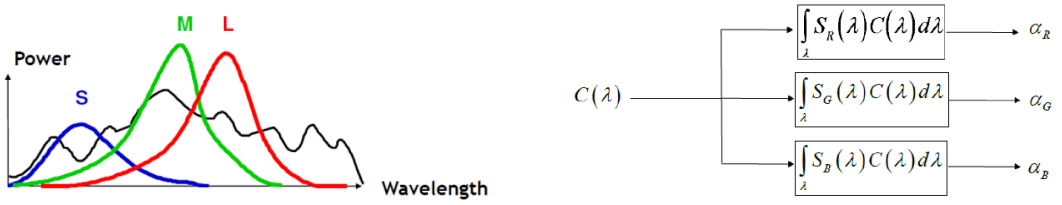

Figure 3.8: Cones as filters of image spectrum

- **RGB:** useful for color monitors
- **CMY:** useful for color printers
- **HSV:** allow to visualize colors into a form closer to the way in which humans describe and interpret them

We now proceed to describe the two models used in this fourth laboratory session.

3.2.1.1 RGB model

This is the most widely used color space in which each pixel of the input image is characterized by Red (645.16 nm), Green (526.32 nm), Blue (444.44 nm) components.

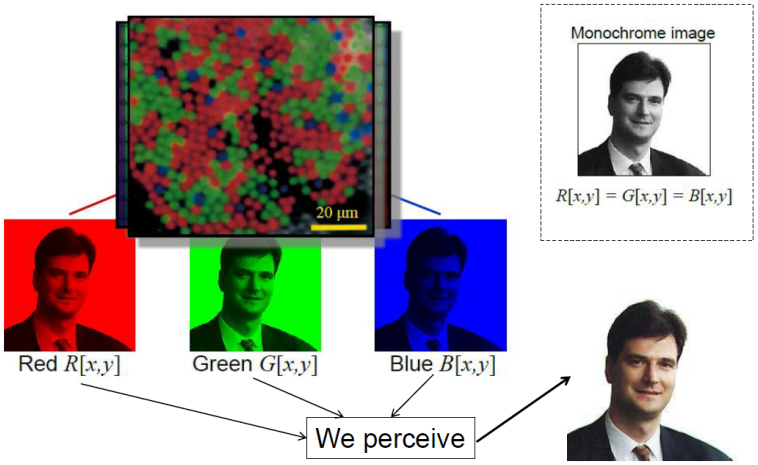

Figure 3.9: RGB decomposition

Available colors for digital image are usually provide by the **RGB cube**[4] whose edges represent the primary colors. With this technique it is possible to describe monochromatic colors as the mixture of the RGB colors.

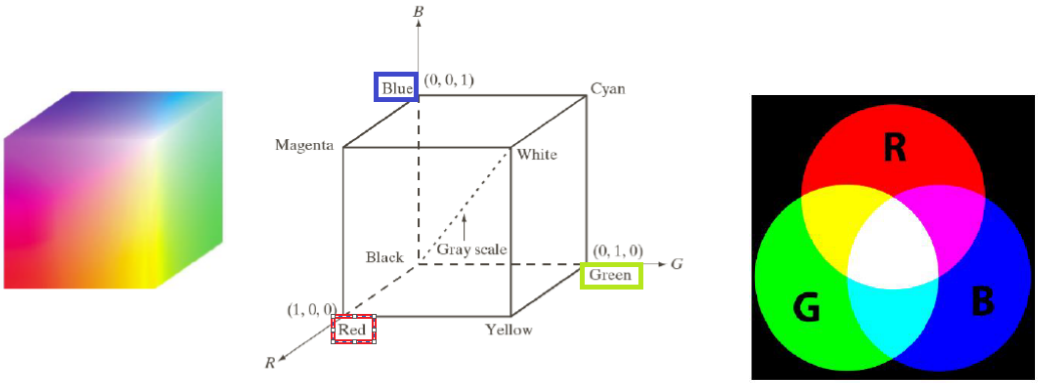

Figure 3.10: RGB cube and additive of primary colors

3.2.1.2 HSV model

This model is an example of the perceptual color space and it allow to quantify the subjective human color perception by means of this three measures:

- **Hue:** property of a color in passing from red to green
- **Saturation:** property of a color in passing from red to pink
- **Value:** property of a color in passing from black to white

This representation is a projection of the RGB color cube into a non-linear chrome angle, radial saturation percentage and luminance inspired value, as follow:

[4] It is a 24-bit representation so it's possible to describe $(2^8)^3 = 16,777,216$ different type of colors.

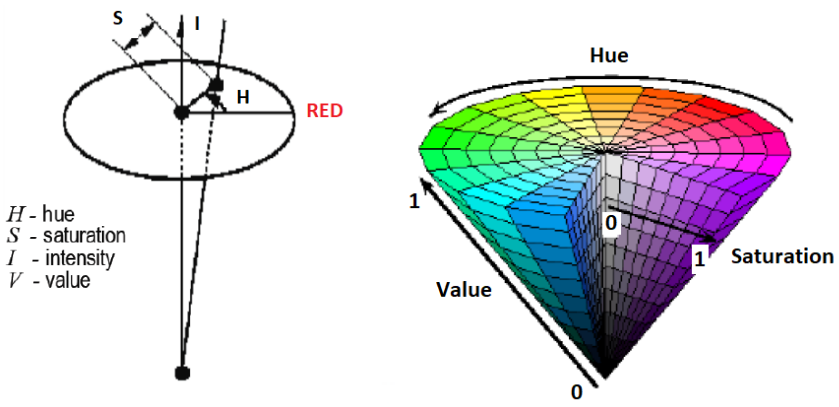

Figure 3.11: HSV representation

3.2.1.3 Comparison between RGB and HSV models

- RGB model:
  - Non-perceptual representation
  - strongly correlated channels
  - each color is obtain using this relation: $color = r * R + g * G + b * B$

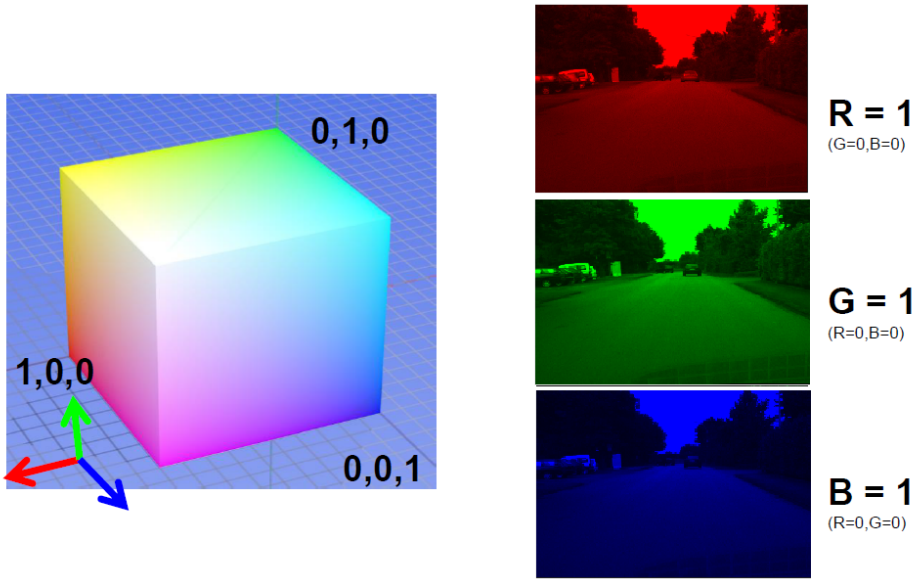

Figure 3.12: RGB decomposition

- **HSV model:**
  - intuitive color space

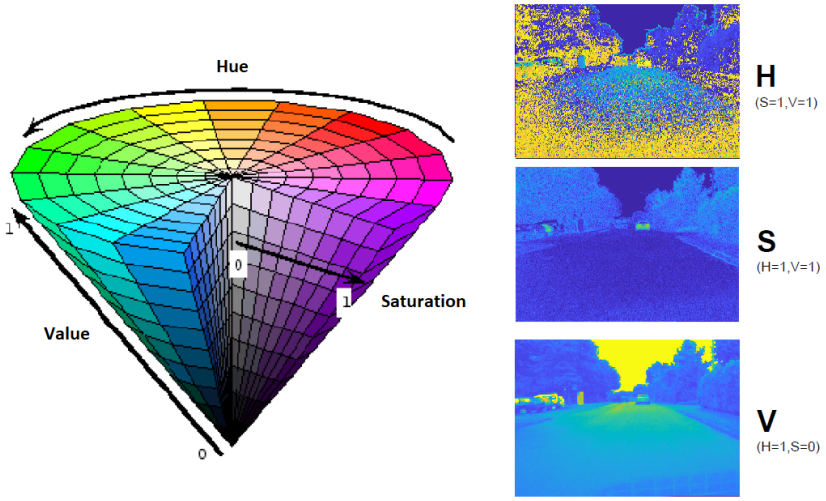

Figure 3.13: HSV decomposition

3.3 Image segmentation based on color

The goal of this technique is to partitioning image into regions based on colors information; pixels inside each of these regions present homogeneous characteristic (i.e. color, intensity, hue value) that are different from the other one.
Considering the simplest case in which the input image I has only one object, for each pixel the following bitmask is performed:

$$B(x,y) = \begin{cases} 1 & \text{if } T_1 < I(x,y) < T_2 \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

where T_1 and T_2 represent two generic threshold values and B is the resultant binary segmented image.
More specifically this method is composed by two main steps:

1. **Seed segmentation:** crude image segmentation based on the gray level distribution, through specific masks, and connectivity of pixels. During this step it is useful to discharge small regions that can be produced by noises, or other factors, in order to obtain a better final result

2. **Region Growing:** merge some small adjacent region into large ones that represent more closely the object

Once the image is segmented into **blobs** we can extract the geometrical properties associated to them. Considering one single blob, described by the nxm B(i,j) matrix, it is possible to define:

- **Area:** total number of pixels occupied by the region

$$A = \sum_{i=1}^{m} \sum_{j=1}^{n} B(i,j) \tag{3.3}$$

- **Centroid:** center of the arbitrary shaped region

$$\bar{i} = \frac{1}{A} \sum_{i=1}^{m} \sum_{j=1}^{n} x \; B(i,j) \qquad \bar{y} = \frac{1}{A} \sum_{i=1}^{m} \sum_{j=1}^{n} y \; B(i,j) \tag{3.4}$$

- **Perimeter:** sum of its border pixels

Chapter 4

Matlab

4.1 Function tree

The code consists of a *main.m* script that runs all the following subfunctions:

1. *RGB_channels.m*
   takes as input an RGB image and displays its Red, Green, Blue values separately each one mapped into grayscale

2. *HSV_channels.m*
   converts the red, green and blue values of an RGB image into a HSV image and displays the results in grayscale mode

3. *histograms.m*
   notes the variation of the RGB components and of the Hue one in the area of the red car

4. *mean_std.m*
   computes the mean and the standard deviation of the Hue values related to the cropped image *car_region.png*[1]

5. *blob_detection_HSV.m*
   detects regions of the image in which the Hue value is inside a the range $m \pm 3 * \sigma$.

[1]The user can also uncomment line 30 and 31 to manually select the desired are.

If more than one region are found the algorithm considers the largest one. The function also prints the values of the area, the coordinates of the boundingBox and the centroid

It is also possible to perform step 3 and 5 using RGB masks, instead of Hue one, by uncommenting the Auxiliary part (line 41-49).

Chapter 5

Conclusion

5.1 Execution time

Profile Summary

Generated 16-Apr-2019 15:25:15 using performance time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
main	1	42.297 s	1.238 s	
blob_detection_HSV	5	22.658 s	3.881 s	
print	21	18.164 s	1.187 s	
graphics\private\alternatePrintPath	21	15.380 s	0.147 s	
HSV_channels	5	10.414 s	1.467 s	

Profile Summary

Generated 16-Apr-2019 15:26:51 using performance time.

Function Name	Calls	Total Time	**Self Time***	Total Time Plot (dark band = self time)
blob_detection_HSV	5	22.658 s	3.881 s	
imagesci\private\pngwritec (MEX-file)	21	3.634 s	3.634 s	
...ut>LocalCallgenerateGraphicsOutput	21	3.155 s	2.962 s	
...ager>@(o,e)subplotlayout(ax,e,fig)	49	4.872 s	2.388 s	
title	83	2.294 s	2.151 s	
...inerFactory$FigurePanelContainerLight (Java method)	42	1.663 s	1.663 s	
HSV_channels	5	10.414 s	1.467 s	

Figure 5.1: Performance time

Without considering functions related to displaying and creating images, *blob_ detection_ HSV()* is the one that spends most of the execution time.

It is important also to highlight that the overall computation time is 42.297s, so it's a time much longer than all the scripts of the previous assignments.

5.2 Results

Among all the possible input images, shown in **Figure 5.2**, we decided to present the outcomes of our code only for the first one.

Figure 5.2: All input images

1. **Display the first image in grayscale and split into RGB and HSV channels:**

Figure 5.3: First image split into RGB and HSV channels

Through these images it is possible to extract color information about the observed landscape in order to find particular range of values that characterize different object.
It's important to remark that *Saturation* and *Value* components are strongly affected by light variations, for this reason segmentation process is based on *Hue* parameter only.

2. **Note the variation of the RGB components and of the Hue one around the red car on the street:**

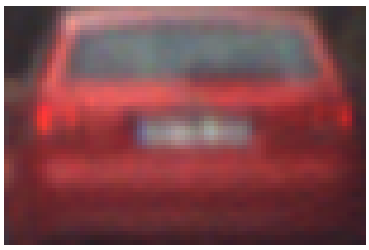

Figure 5.4: Desired red car

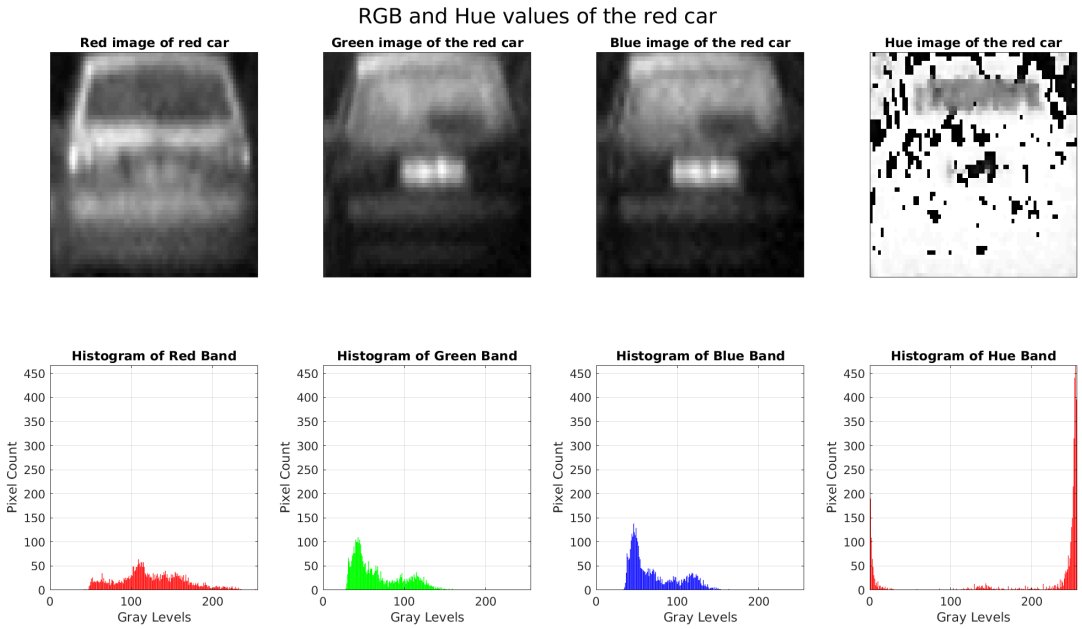

Figure 5.5: Histograms of the RGB and Hue components around the considered car

From the **Figure 5.5** we can clearly see that RGB values are not indicated to detect specific object, because of their flatness and their high dispersion instead of *Hue* value that exhibits a high peaks located around 0 and 1 values.

3. **In the area of the car compute the mean value and the standard deviation of the Hue component**

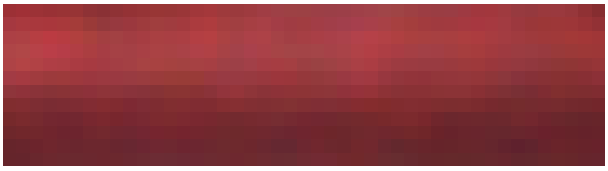

Figure 5.6: Sample of the car body

As previously mentioned, *Hue* component is the best choice, between all the HSV and RGB parameters, for detecting our subject.
Before computing the mean and the standard deviation of this cropped image, it is necessary to delete the gap between the elements near to zero and to unit values. This step is necessary because the *Hue* component is an angular value with period [0,1] so the two ends collapse into the same point. For this reason all the Hue values lower than **0.1** are shifted by a unit.

Considering **Figure 5.6** as input we obtain the following results: mean value approximately equal to **0.9.8997** and standard deviation equal to **0.0063**. Note that if we had no shifted the *Hue* values the resulting standard deviation would have been **0.2465**.

4. **Segment the red car by thresholding the Hue component**

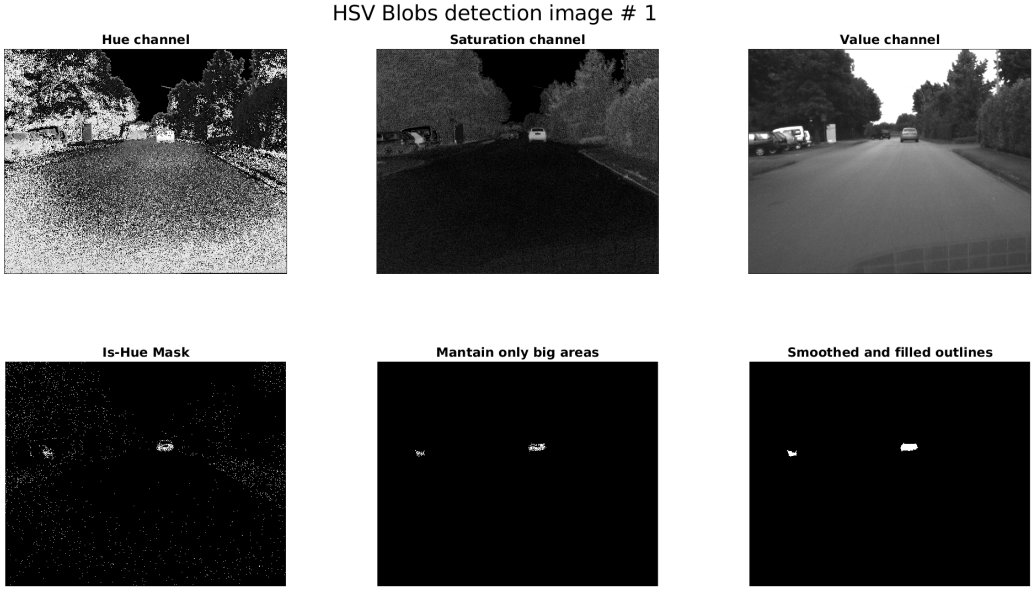

Figure 5.7: Blob detection applying a filtered Hue mask

From the first top right image it is possible to notice that, beneath the red car region presents high Hue values, presents widespread values throughout the landscape. After cutting off the undesired intensities, a Gaussian filter, with kernel dimension $7x7$ and $\sigma = 0.5$, was applied in order to reduce the noise.

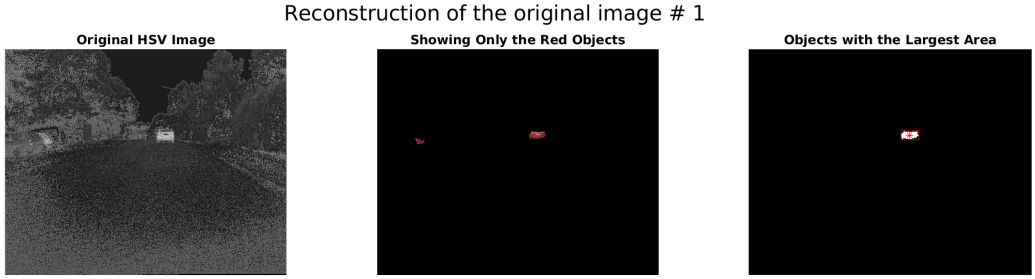

Figure 5.8: Detection of the desired subject

The second figure on the above subplot shows the input image filtered with the bitmask previously provided, while the right one presents the geometrical properties of the output, in particular the bounding box and the centroid.

The last image shows the up mentioned properties on the initial RGB image.

Figure 5.9: Final image