Computer Vision Laboratory Report n.4

COLOR-BASED SEGMENTATION

April 15, 2019

Claudio Curti ID: 4216203
Nicola De Carli ID: 4198668
Alberto Ghiotto ID: 4225586
University of Genoa

Contents

Chapter 1

Introduction

1.1 Defining the objective

This project aims to gain a clear understanding of a rich and complex experience which is color perception. In order to achieve this intention this experiment will exploit the concept of color space, eventually orienting the main focus of this work on color-based segmentation relying on hue thresholding by tuning parameters such as mean value and standard deviation. Once the desired color is detected in the image, it will be possible to mark it through the so called "bounding box" and the corresponding centroid.

Figure 1.1: Main image used for the experimentation

Chapter 2

Theoretical Background

Colorspaces, such as RGB, CMY, HSI/HSV, are normally invented for practical reasons, in order to adapt to the current application. With the purpose of segmenting the image, the obvious choice was to focus on HSV in order to easily perform the color analysis thanks to its convenient representation and particularly the hue channel turns out to be very useful. The hue channel represents the tone of the color and its values are represented as angles that vary from 0° (primary red), passing through 120° (primary green), then 240° (primary blue), and finally again to red at 360°. It is particularly useful to detect a particular color because in theory its value is independent from luminosity variations.

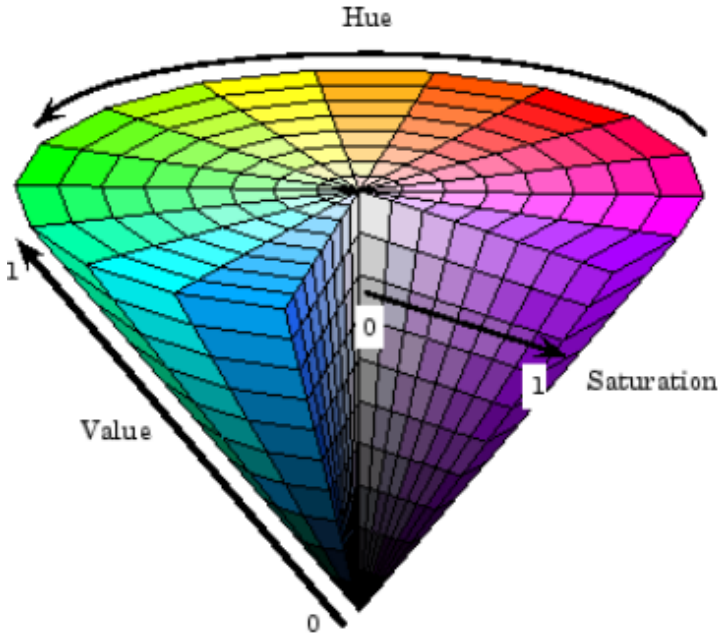

Figure 2.1: Hue-Saturation-Value domain

Chapter 3

Implementation and Results

The code is organized using a script, which loads the RGB images in the workspace and then calls all the developed functions. In order to carry out all the required manipulation the following functions were developed:

- **convertAndShow**: which simply converts the image from RGB to HSV domain and show separately the three different channels H, S and V.
- **hueID**: which computes the mean value and the standard deviation of the hue values in a portion of the image.
- **colorSeg**: which segments all the images using as parameters the mean and the standard deviation found previously.
- **dispCBB**: which shows the binary images corresponding to the segmentation along with the centroid and bounding box of the objects detected.

3.1 Converting and displaying in different colorspaces channels

In the first part the images are converted in grayscale and splitted in the three channels of RGB: red, green and blue and in the three channels of the HSV: hue, saturation and value (or intensity). In order to perform the conversion from RGB to HSV the MATLAB function "rgb2hsv" was used. The results are shown in the following images where it is possible to see how much the information contained in the RGB channels is correlated, while, in the HSV channels, the information contained in the various channels concerns different aspects of the images.

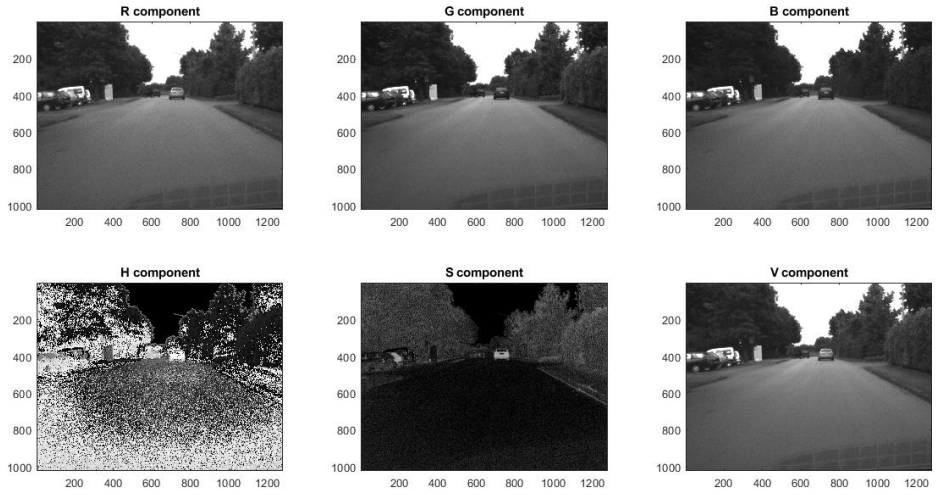

Figure 3.1: RGB and HSV color domains

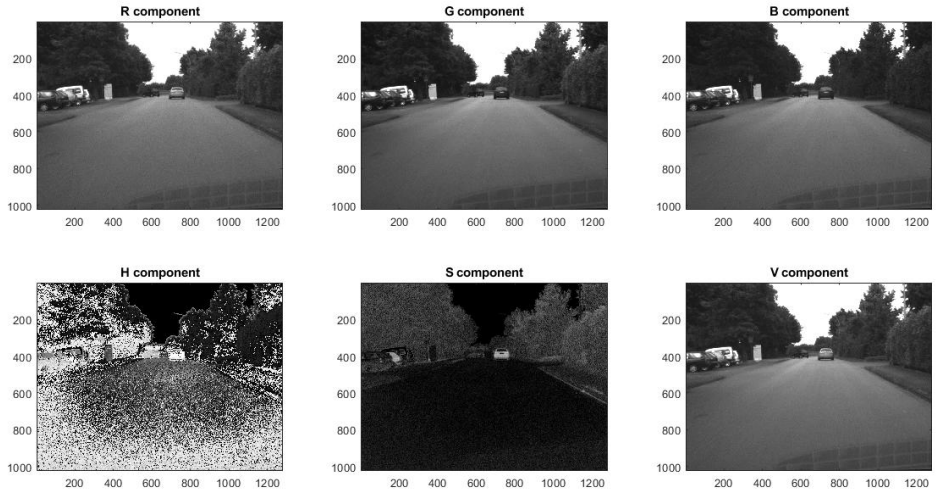

Figure 3.2: RGB and HSV color domains

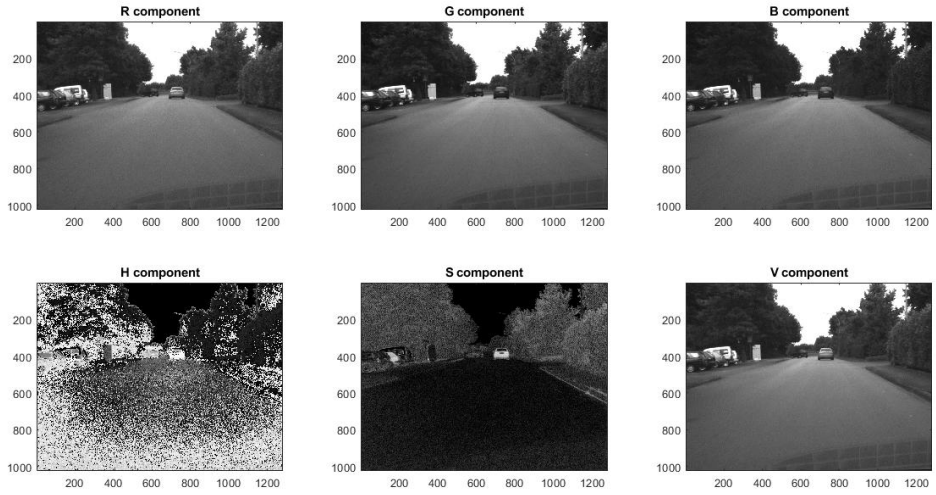

Figure 3.3: RGB and HSV color domains

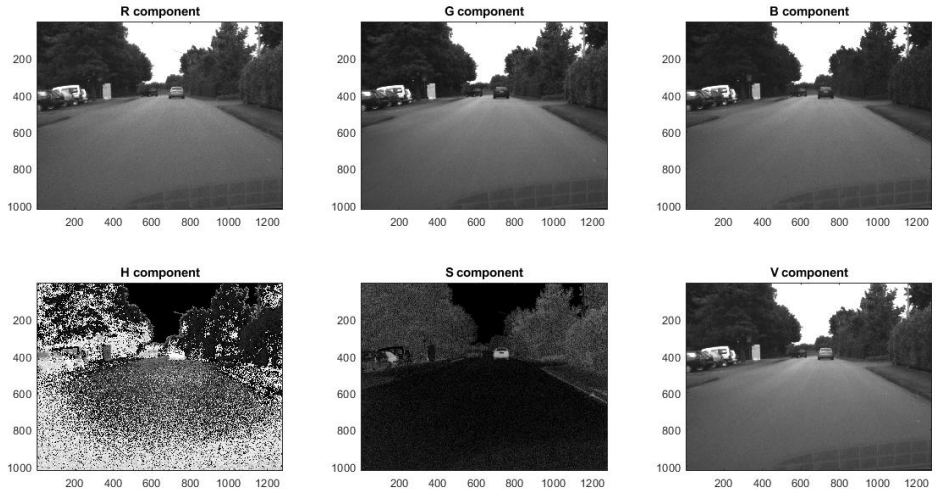

Figure 3.4: RGB and HSV color domains

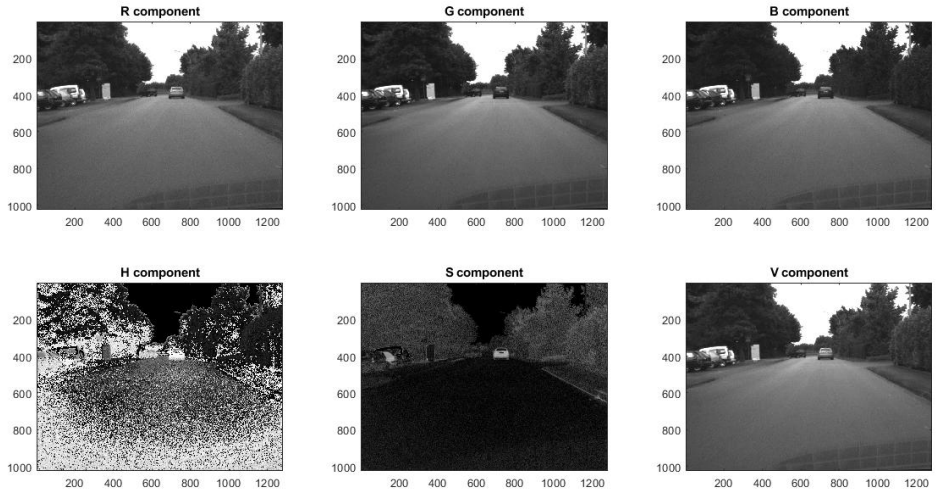

Figure 3.5: RGB and HSV color domains

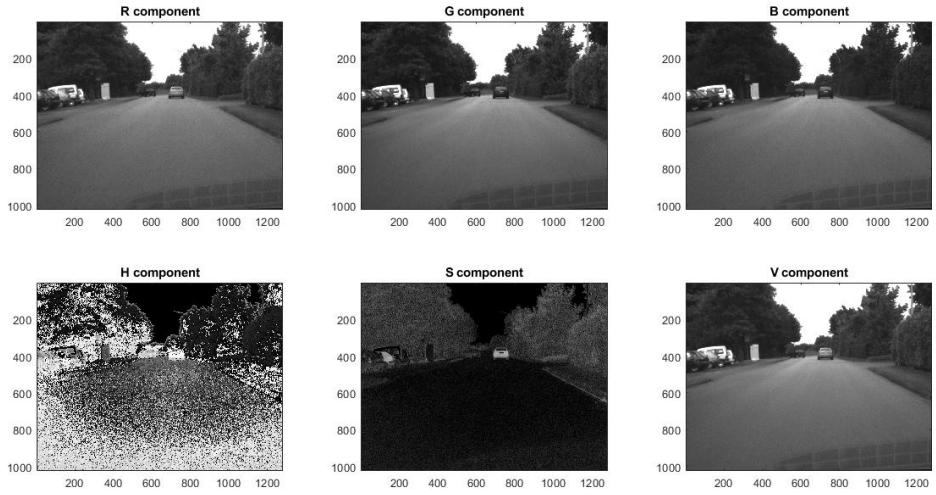

Figure 3.6: RGB and HSV color domains

3.2 Hue value detection

In the "hueID" function the hue mean value and the hue standard deviation values of a given rectangular area are detected. Due to the circular nature of the hue channel this task is a little tricky and the solution methods not so straightforward. Since in MATLAB hue values are between 0 and 1, in the case in which it is required to detect the hue values in an area in which pixels are red, the values could vary around a little more than zero and a little less than one. Therefore the mean would be around 0.5 which makes no sense since it is the most distant value from red. Luckily, there exists various techniques to avoid this problem, such as exploiting the sum of sines and cosines and then reverting with atan2 or the other way consists in, as used in this experiment, shifting the high half of the interval, the values between [0.5,1], of -1, resulting in values centered around zero. This second technique, in contrast with the first, works only for some ranges of values, in fact, it simply moves the problem from detecting colors near the red value to detecting colors whose hue value is around 0.5. To detect the standard deviation, instead, the only way is the same as the second technique used to detect the mean. This function only displays the rectangle used for the parameters measurement, as it can be seen in Figure 3.7.

Figure 3.7: Red car area

3.3 Color segmentation

Once the mean value and the standard deviation were computed, it has been possible to proceed with the color segmentation which, as required from the text, allows to recognize the two red cars inside the picture from the rest of the objects that belong to the image. In order to obtain a smoother image allowing an easier future segmentation, the function performs a prefiltering of the original image by using a gaussian filter (with $\sigma = 3$ and support = 19). After that the image is converted to HSV domain and a mask is applied to the hue component to extract the desired feature. The range of the mask varies between m - 3σ and m + 3σ, keeping into account the circularity of the color, outlined in the previous section. Since the obtained image showed a lot of undesired details, the following phase, the centroid and bounding box detection, would have been difficult to achieve. For this reason, techniques of morphological image processing have been applied. In fact, initially the erosion is applied to remove small details, then dilation to restore the size of the objects that are meant to be detected, all this using as structuring element a square. Eventually, the function "bwareaopen" is applied to consider only elements whose area of pixels is greater than a chosen threshold. The results of this function are displayed at the end of the following section together with the centroids and the bounding boxes of the objects detected.

3.4 Centroid and bounding box

The "dispCBB" function takes as input a segmented image and the original image and plots on both of them the centroids relative to the objects detected and encloses the objects in rectangles. It exploits the function "bwLabel" which takes as input a binary image and, by setting all the bit belonging to a connected element to a specific value, puts labels on the connected elements present in the binary image. Then, for each connected element, all the rows and columns indexes of the relative pixels are stored in arrays and the means of these vectors are computed in order to find the coordinates of the centroid of the element. Then, by finding the maximum and minimum values of the arrays of the indexes, the vertices of the bounding box are computed for each element. The final result obtained for each picture is displayed in the following images.

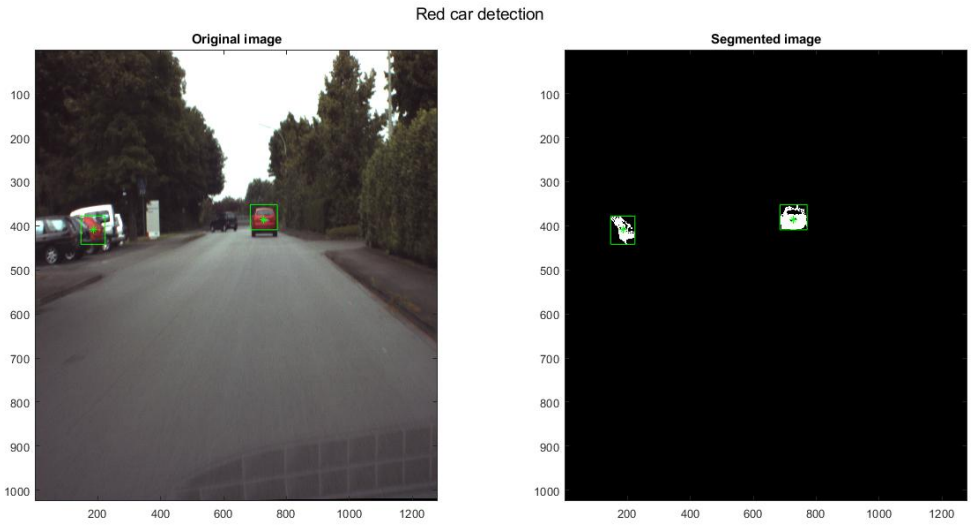

Figure 3.8: Red car detected in frame 1

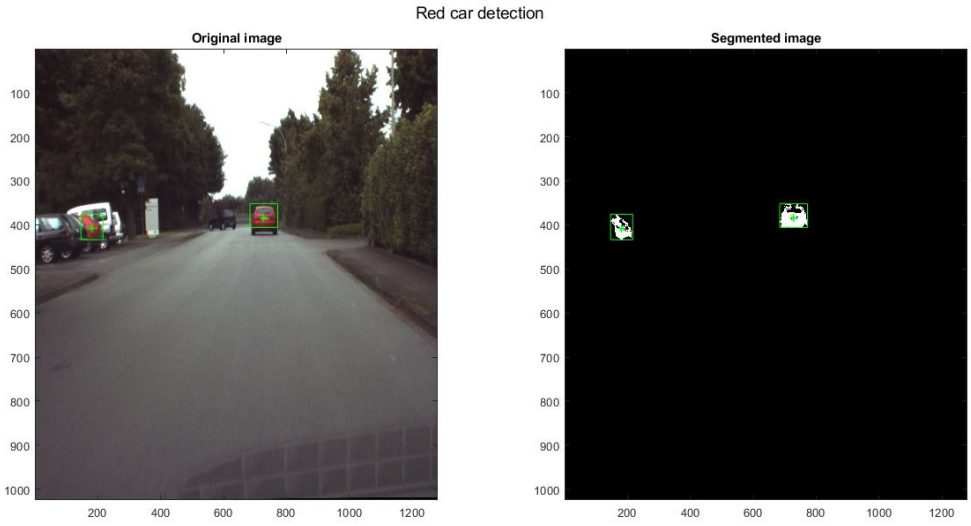

Figure 3.9: Red car detected in frame 2

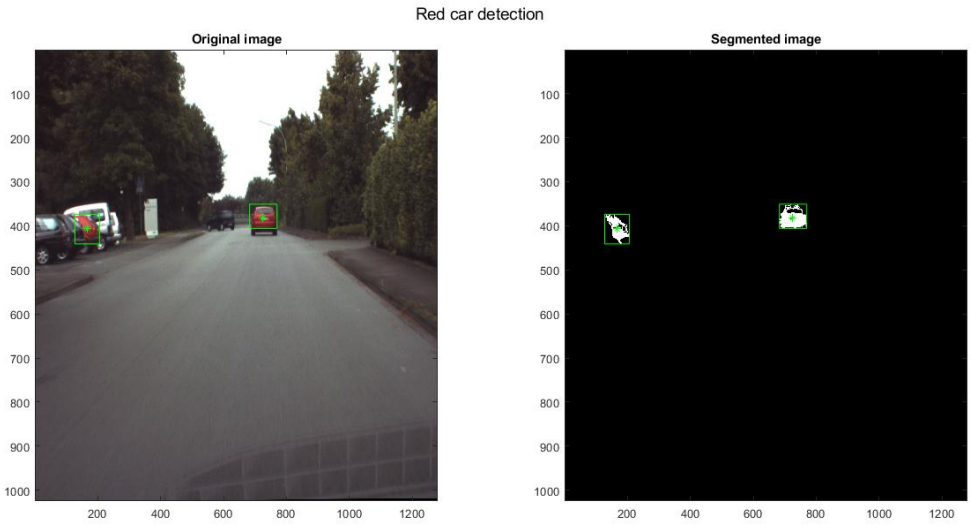

Figure 3.10: Red car detected in frame 3

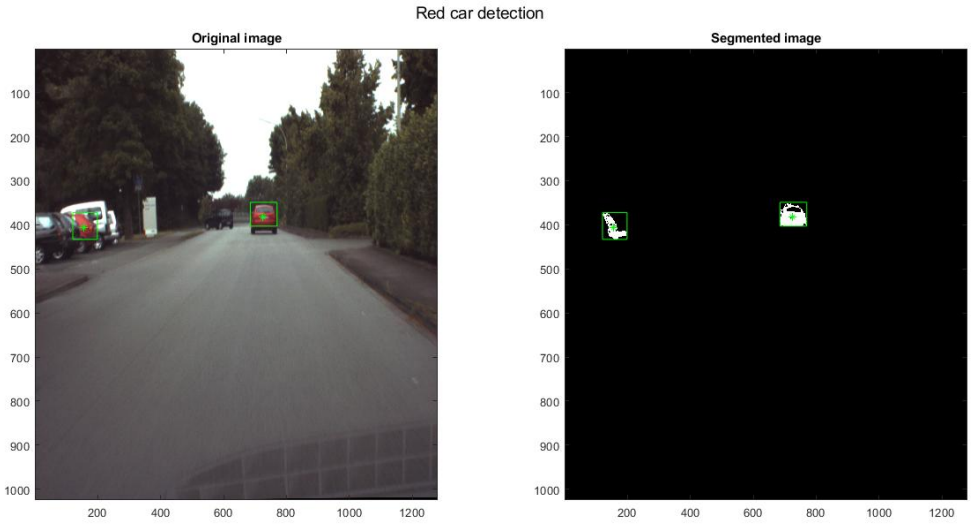

Figure 3.11: Red car detected in frame 4

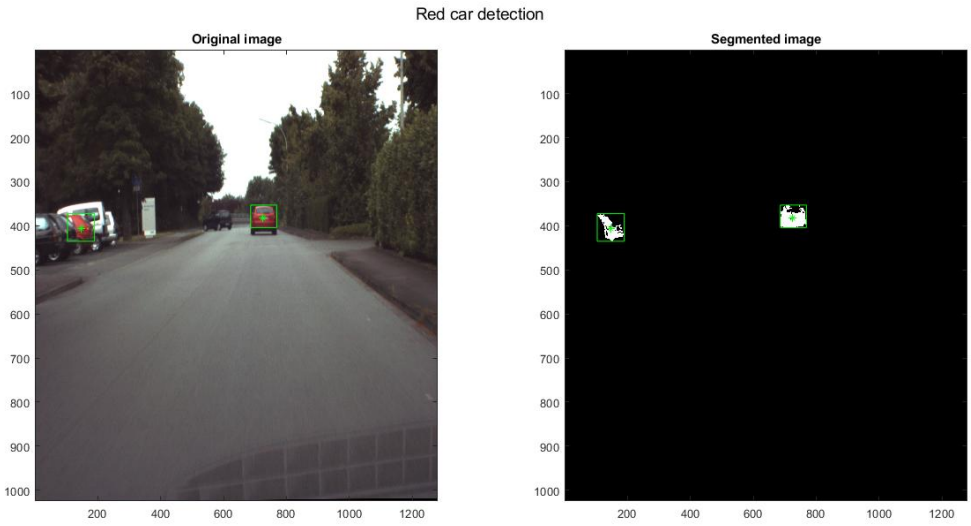

Figure 3.12: Red car detected in frame 5

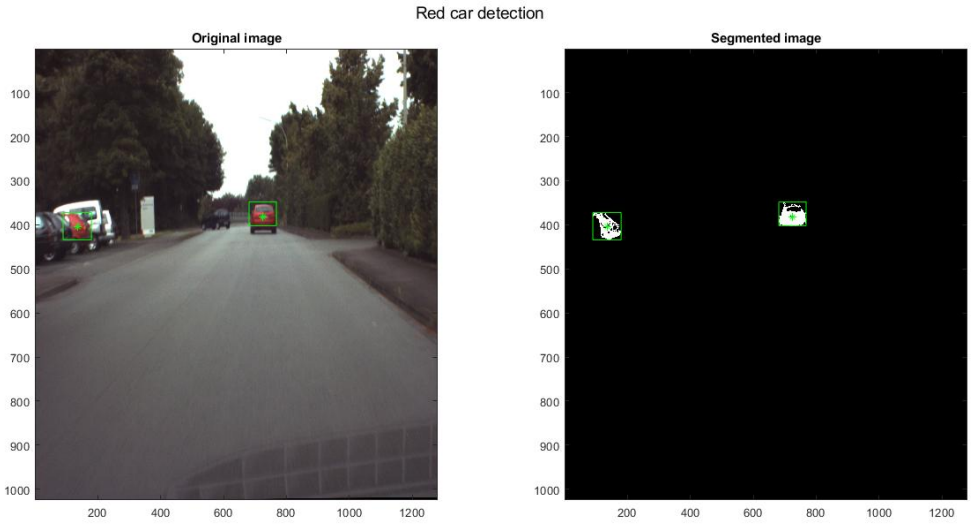

Figure 3.13: Red car detected in frame 6