Computer Vision Laboratory Report n.4

COLOR-BASED SEGMENTATION

April 14, 2020

Chiara Terrile ID: 4337786
Giulia Scorza Azzarà ID: 4376318
University of Genoa

Contents

Chapter 1

Introduction

1.1 Defining the objective

The first part of this laboratory deals with the color perception, while the second part deals with the scale-space blob detection. The main objective is to perform a blob detection. This can be done in several different ways, of which we have considered one based on color segmentation and another one based on the scale-space.
The goal of color-based segmentation is to identify homogeneous regions in a color image, that represent objects or meaningful parts of objects present in the scene. In order to achieve this goal the experiment exploits the concept of color space; in our case the main focus is oriented on the hue component and on 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.
Regarding the application of scale-space analysis, the objective is to implement a method that localizes circular patches (blobs) in a given image. To achieve this goal it's used the Laplacian filter to detect step-like patterns across the scale-space and the final blob location is reported based on local maxima, detected in the Laplacian filter response across scale-space.

Chapter 2

Theoretical Background

2.1 Color-Based Segmentation

A color space is a specific organization of colors. For the purpose of this laboratory, the better choice was to use the HSV color space, in order to easily perform the color analysis thanks to its convenient representation. Hue Saturation Value (HSV) indicates both an additive method of composition of colors and a way of representing them in a digital system. About the three parameters, the Hue varies conventionally starting from primary red at 0°, passing through primary green at 120° and primary blue at 240° and then returning to red at 360°; then Saturation represents the intensity and purity of the single hue, while Value is an indication of its brightness.

2.2 Scale-space blob detection

The center of the flat region of a step-like feature (in our 2D case, the blobs) should result in a local maxima in the filter response across scales. Detecting such a peak enables us to localize the center of the corresponding blob.
This involves searching a 3D neighborhood around a pixel in the combined scale-space to ensure that it is indeed a local minima.
Once a local maxima is found, the size (radius) of the corresponding blob can be determined by looking at the expression for the Laplacian as the second derivative of the Gaussian kernel. The radius of the corresponding blob is given in terms of the so called characteristic scale (σ) as:

$$r = \sqrt{2}\sigma$$

We define the **characteristic scale** as the scale that produces peak of Laplacian response.

Chapter 3

Implementation and Results

The code is composed of two **main** functions, one for the six images representing the black and red cars crossing the road (**main_cars**) and the other one for the image 'sunflowers.jpg' (**main_sunflowers**).
In both main scripts are called several functions, in particular the first five functions are used for the first main and the other four for the second one, in order to perform the color-based segmentation of the required objects inside the images.

All the functions that we have implemented are shown below:

- **RGB_HSV.m:** which displays the six images splitted in the three RGB channels and in the three HSV channels
- **hsv.m:** which compute the mean value and the standard deviation of the Hue component
- **segmentation.m:** which returns the segmented image
- **largestBlob.m:** which detects the biggest blob in the image
- **plottingSeg.m:** which plots the segmented image with the detected blob
- **imgScale.m:** which computes the Laplacian response of the image and displays it in function of the scale
- **circlesDetection.m:** which detects all the circles in the image
- **biggestCircleDet.m:** which detects the biggest circle in the image
- **showAllCircles.m:** which displays all the circles in the image given the positions of their centers

3.1 main_cars

In the first main function we have loaded all the six images and plotted them (see the result in Figure 3.1); then we have applied the function **RGB_HSV** to perform the first part of our experiment.

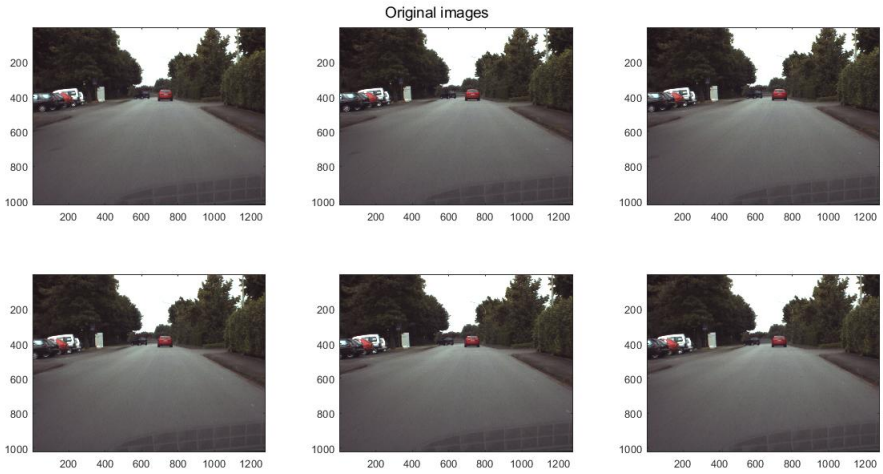

Figure 3.1

In the second part the aim has been to detect a black car and a red car in all the images. In order to detect the black car, we have selected the area which contained the car (img_z) and we've applied the function **hsv** to this area, obtaining the mean value and the standard deviation of the Hue component.
These values (m and s) have been used to compute the thresholds for the segmentation process; we have chosen the threshold (m-s, m+s) to select the area of the black car.
For the red car, instead, we've used the given precise thresholds (0.97, 1) without passing through a selected area as we did for the black car.
After detecting the two cars for all the images, the results have been plotted with the function **plottingSeg**.
In the following pages are explained all the functions that we have used.

3.1.1 RGB_HSV

This function takes as input the image we want to split in the RGB and HSV channels and returns as output the image in RGB (img_rgb) and the one in HSV (img_hsv).
To convert the image from RGB format into HSV format we have used the MATLAB function **rgb2hsv**.
In the following pages are shown all the results from Figure 3.2 to Figure 3.7.

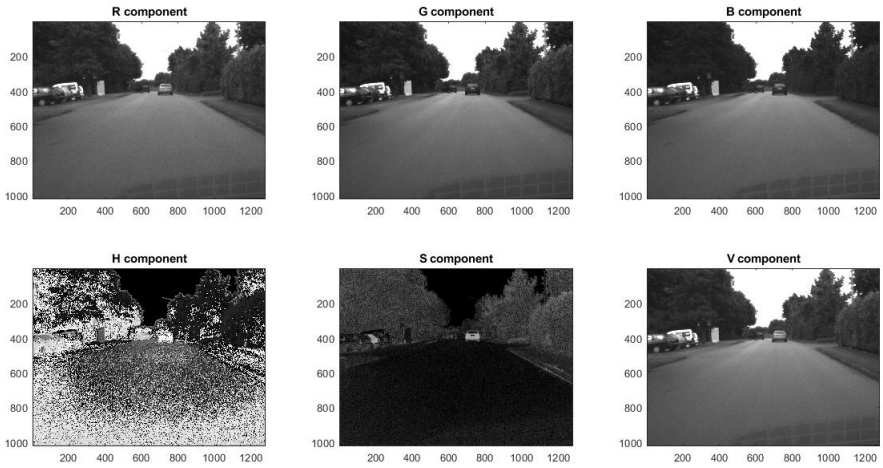

Figure 3.2: Frame 1

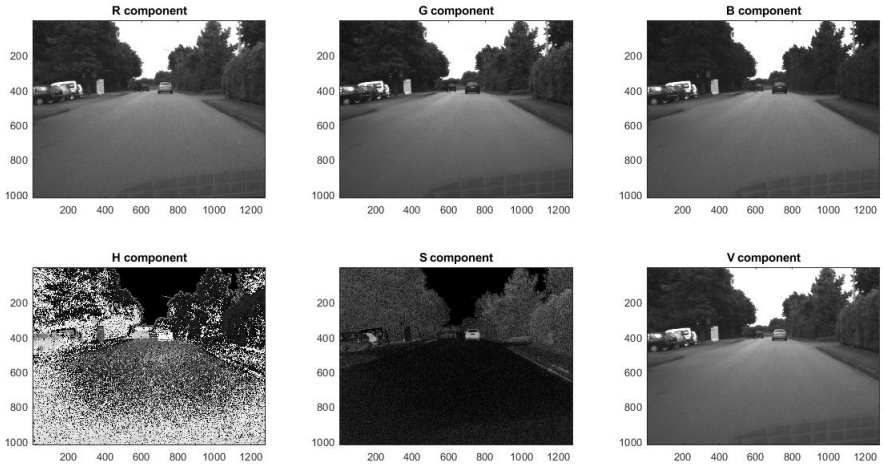

Figure 3.3: Frame 2

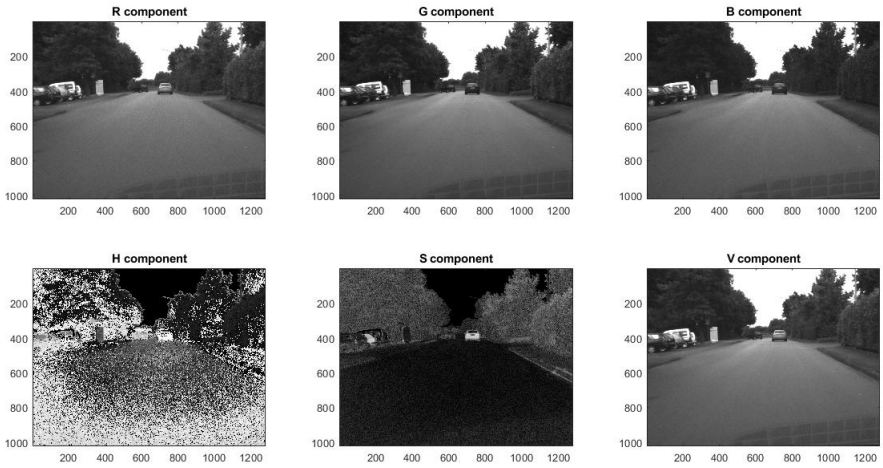

Figure 3.4: Frame 3

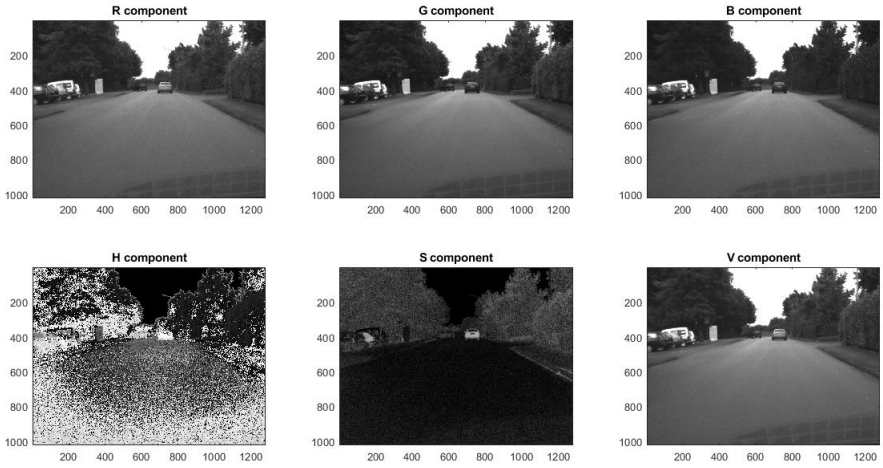

Figure 3.5: Frame 4

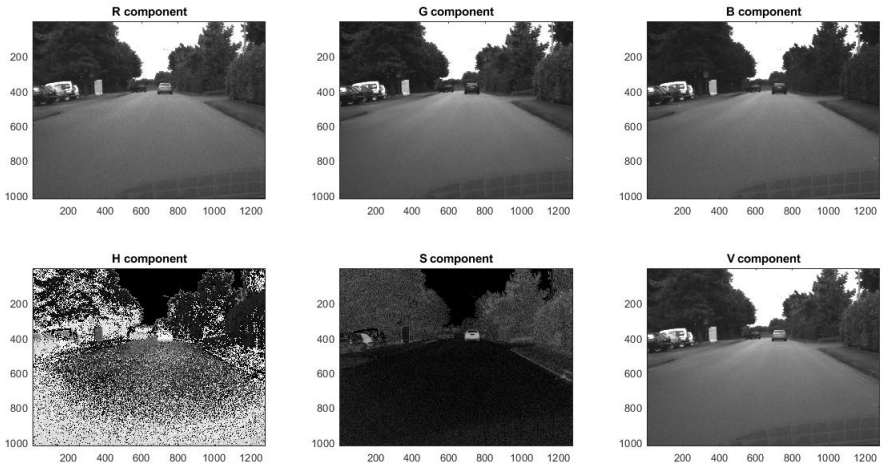

Figure 3.6: Frame 5

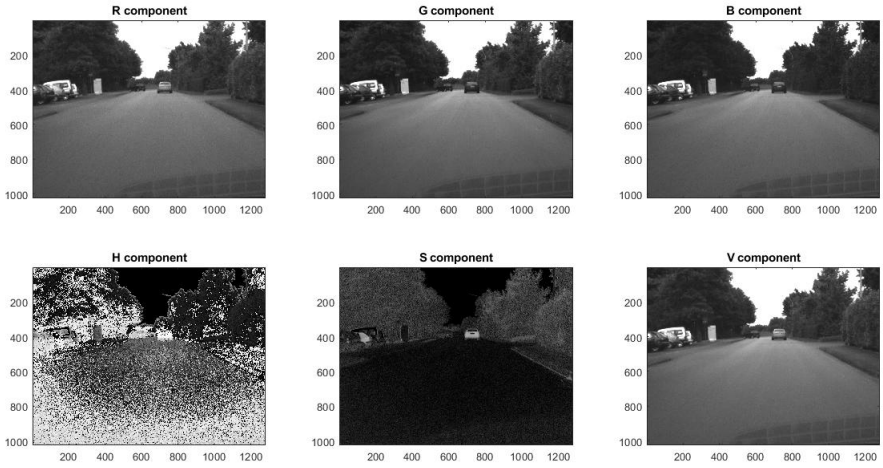

Figure 3.7: Frame 6

3.1.2 hsv

This function takes as input the image in HSV format (img_hsv) and returns as output the mean value and the standard deviation of the Hue component.
To obtain m and s, we've used two MATLAB function, that are respectively **mean2** and **std2**, to which we've passed as input the Hue component h.

3.1.3 segmentation

This function takes as input the image in HSV format (img_hsv) and the threshold values (th1 and th2) and returns as output the selected blob (BinaryImage) and the segmented image (seg).
After the segmentation of the image, using a mask with our thresholds, we have called the function **largestBlob** to select the biggest blob in the segmented image.

3.1.4 largestBlob

This function takes as input the segmented image and the number of blobs that we want to extract (numberToExtract) and returns as output the biggest blob (BinaryImage).
For the implementation we have applied the MATLAB function **regionprops** to the segmented image in order to obtain information about the areas of the blobs.
Then we've ordered all the areas in a descendent order using the MATLAB function **sort**.
Finally, according to the number of blobs to obtain, the function selects the first n-blobs.

3.1.5 plottingSeg

This function takes as input:

- the original image (img)
- the selected blob containing the black car (binaryImageB)
- the segmented image for the black car (segB)
- the selected blob containing the red car (binaryImageR)
- the segmented image for the red car (segR)

In this function are computed the coordinates of the centers of both blobs in order to plot the exact position of them in the image. We have used again the MATLAB function **regionprops**, to detect also the BoundingBox of the two blobs.
The output of this function is a figure containing the segmented image with the detected blob and also the original image with the detected blob, for both cars.
In the following pages are shown all the results from Figure 3.8 to Figure 3.13.

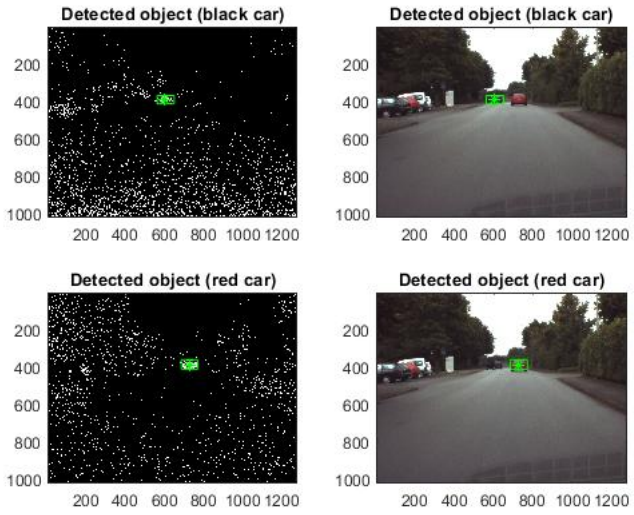

Figure 3.8: Frame 1

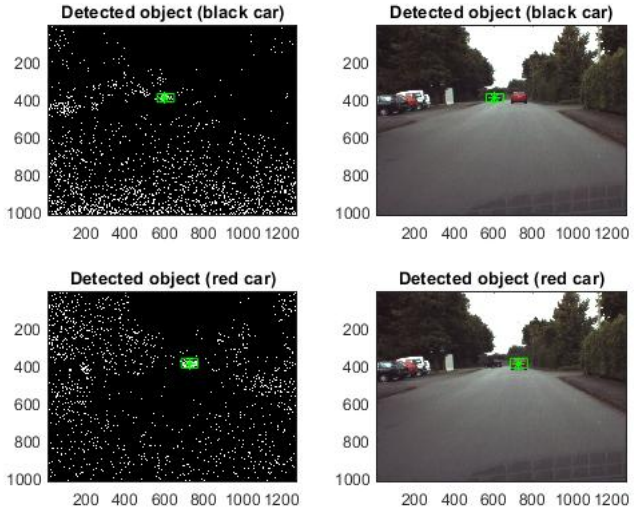

Figure 3.9: Frame 2

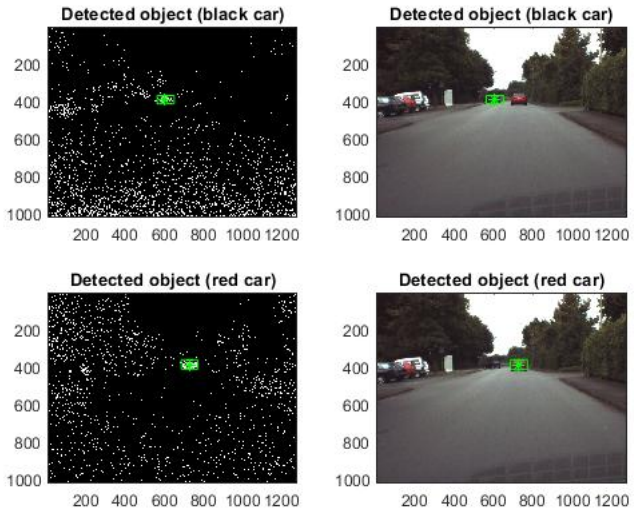

Figure 3.10: Frame 3

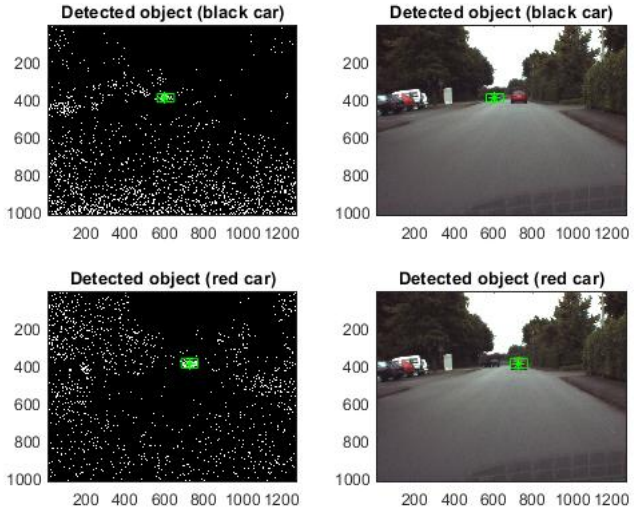

Figure 3.11: Frame 4

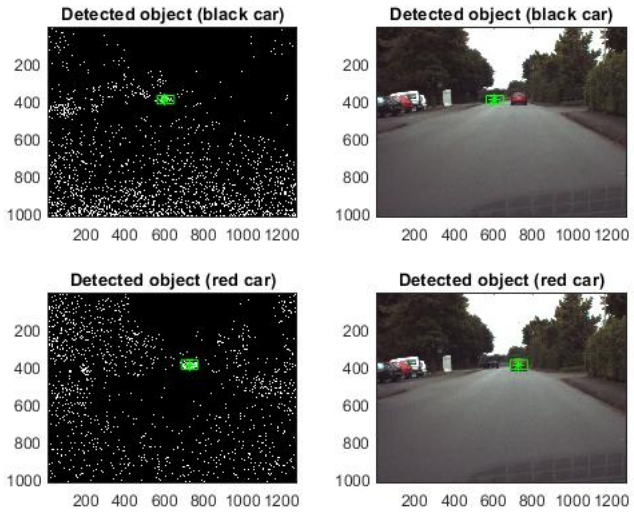

Figure 3.12: Frame 5

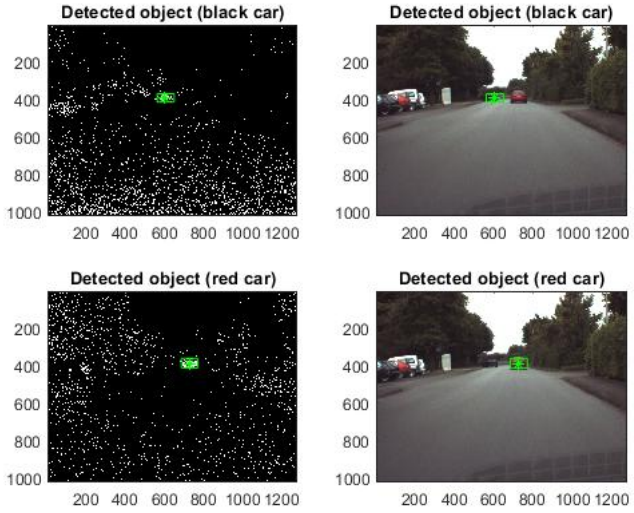

Figure 3.13: Frame 6

3.2 main_sunflowers

In the second main function we have loaded the image 'sunflowers.png' and we have selected two areas containing two flowers in particular, identified as 'flower1' and 'flower2'.
First of all we have applied the function **imgScale** to obtain the Laplacian response.
Then we have applied the function **biggestCircleDet** to both flowers in order to compute the characteristic scale for each flower and the coordinates of their centers.
Finally, using the function **showAllCircles**, we have plotted the original image with the two flowers detected, passing as input the coordinates of the centers and the scale (sigma) multiplied for $\sqrt{2}$, according to the relation between the characteristic scale and the radius. In the following pages are explained all the functions that we have used.

3.2.1 imgScale

This function takes as input the image we want to filter with the Laplacian filter (img) and a string to be inserted in the title of the subplot.
In this function the standard deviation (sigma) is set to the initial value of 1, the number of scales is set to 10 and the increment to 1.5.
In every for-loop the Laplacian filter is applied to our image with a certain sigma, which represents the scale, as it is shown in Figure 3.14 and Figure 3.15 below.

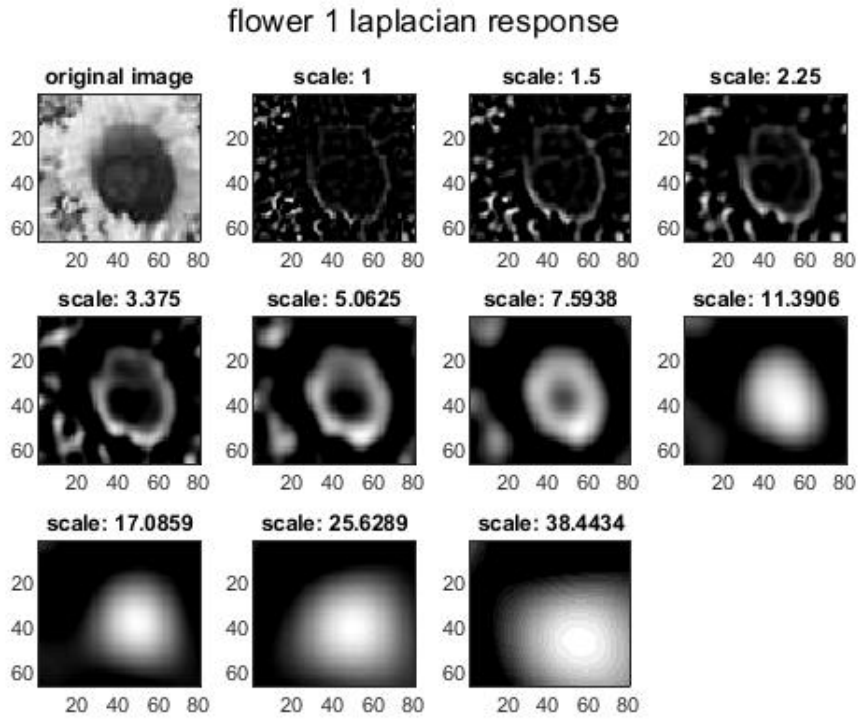

Figure 3.14: The increment value is set to 1.5.

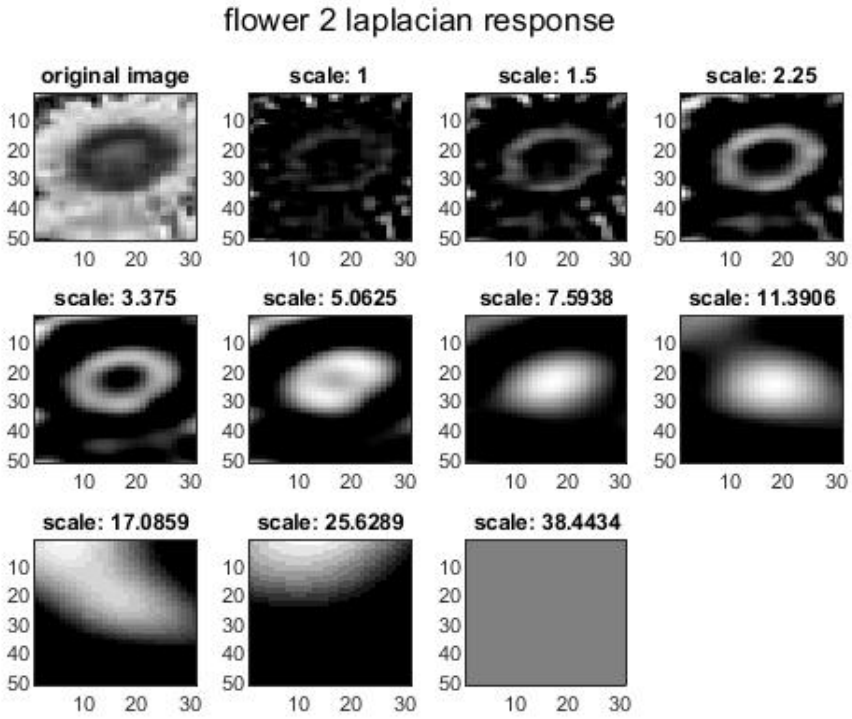

Figure 3.15: The increment value is set to 1.5.

3.2.2 circlesDetection

This function takes as input the image in which we want to detect circles and returns as output:

- **val**: which is a column vector with all the scales of the circles
- **col**: which is a column vector with x coordinates of circle centers
- **row**: which is a column vector with y coordinates of circle centers

In this function are detected all the blobs and by using the MATLAB function **find** are detected also their radii and the coordinates of the centers.
Finally they are displayed through the MATLAB function **showAllCircles**.
Our results are shown in the following page in Figure 3.16 and Figure 3.17.

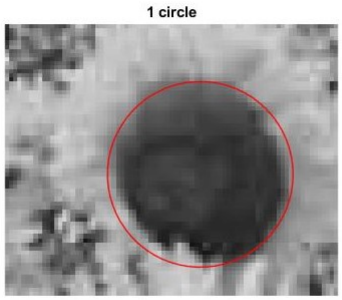

Figure 3.16: flower 1

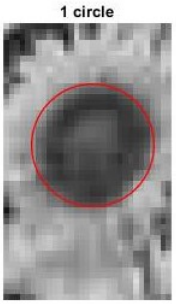

Figure 3.17: flower 2

3.2.3 biggestCircleDet

This function takes as input the original image (img) and two integers extC and extR, which are the values to add to the coordinates obtained with **circlesDetection** in order to re-adapt the coordinates of the smaller images (flower1 and flower2) to the bigger one (the original image 'sunflowers.png').
As output this function returns:

- **colNew**: which is the x coordinate of the center of the biggest circle
- **rowNew**: which is the y coordinate of the center of the biggest circle
- **sigmaNew**: which is the scale of the center of the biggest circle

In this function is called the function **circlesDetection** in order to get the coordinates and the sigma of the circles in a certain image; then with the MATLAB function **sort** the val vector is ordered to find the biggest value. Once found the biggest sigma, are computed also the coordinates of the corresponding circle.
The final step is to adapt the values found before to the starting image, using extC and extR to get colNew and rowNew.

3.2.4 showAllCircles

This function takes as input several values:

- **I**: which is the image on top of which w want to display the circles
- **cx and cy**: which are the column vectors with x and y coordinates of circle centers
- **rad**: which is a column vector with radii of circles.
- **color**: which is an optional parameter specifying the color of the circles
- **ln_wid**: which is the line width of circles

This function simply displays the circles, whose coordinates and radii are given, on the image I, as it is shown in the following Figure 3.18.

Figure 3.18: Detecting circles in **sunflowers.jpg**