Electronic and Telecomunication Engineering
and IT Engineering Department

Image Processing

Coin Detection and Counting

Professor
Pedro Mendes Jorge

Student
Ferro Enrico
Number 44566

Academic Year 2017/2018

Contents

1 Introduction **4**

2 Software description **5**
2.1 User's use and results 5
2.2 Additional functions 5

3 Implemented algorithm **6**
3.1 Histogram analyzing of gray-scale image and threshold value selection 6
3.2 Morphological operations 7
3.3 Labeling operations and contours recognition 8
3.4 Coins recognition 8

4 Libraries used in the code **10**
4.1 OpenCV 10
4.2 NumPy, Scipy and Matplotlib 10
4.3 Math, bwLabel and psColor 11

5 Conclusions **12**

6 References **13**

Chapter 1

Introduction

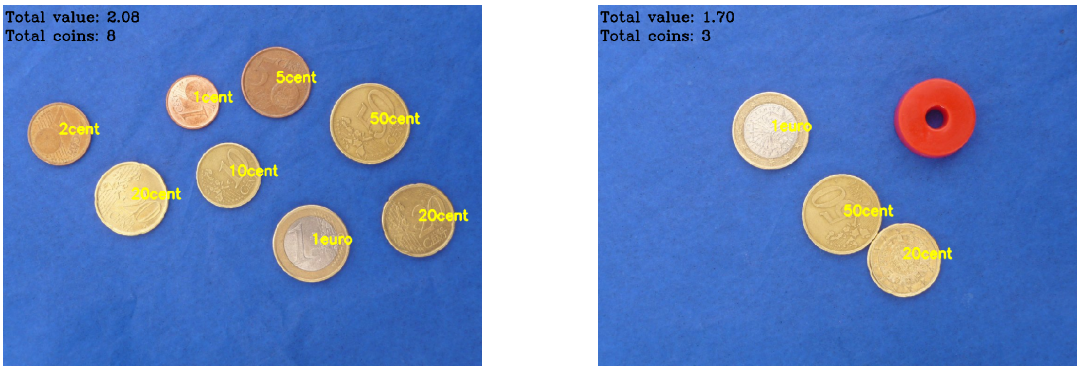
The software described in this report is able to automatically count the amount of coins placed upon a table.
It is supposed to have a camera placed on a tripod and it have the sensor plane parallel a table with a clear and homogeneous surface where coins and other objects are placed.
The software is able to distinguish and recognize only coins even if two of them are in contact, to ignore the other objects present in the plan and to work even in conditions of different shadow.
The application was written in Python and was used the specific image processing library called OpenCV. The recognize algorithm is based about on 9 images used as training for implemented the various cases described.
For use this software the user need: python 3.5 and some libraries installed (openCV, numpy and scipy), and finally two python scripts (*bwLabel.py* and *psColor.py*) that will have to be placed in the same folder of the main script (*coin_det.py*).

Chapter 2

Software description

2.1 User's use and results

The functioning of the program is extremely simple: starting the script coin_det.py, the user must enter the name of the image (including the file extension) that he wishes to parse and, after inserting it, the software performs the necessary steps for the recognition and coin counting and creates a window where the user can view the original image with the name of each coin written above it, the total number of coins and the euros value in the picture. Finally, to close the program, the user can press any key on the keyboard and the software will destroy the generated windows.
Figures 2.1(a) and 2.1(b) show two examples of final result. In both figures you can see how all coins are recognized and counted correctly, furthermore in figure b it can be seen that any other object on the plane does not interfere with the final result.



(a) (b)

Figure 2.1: *Examples of final result*

2.2 Additional functions

The previous section describes the basic use that the user can make of this application, but if you open the code you can see how there are some commented rows. If you remove the hashtag from these, you can see the data and images that describe the step-by-step algorithm. These features are described in detail in the next chapter.

Chapter 3

Implemented algorithm

This chapter describes the project choices made and the various steps that the software performs to recognize coins in the images.

3.1 Histogram analyzing of gray-scale image and threshold value selection

When the user inserts the name of the picture that he wants to use, the system read this image in two ways: with its normal color, and it is saved in the variable *srcColor*, and with gray-scale, variable *src*.
Starting from the grayscale image the software creates a histogram that describes how the grey is distributed in the image, an example can be seen in the figure 3.1.
In the histogram you can recognize two distinct areas, they are shaped like a Gaussian curve:

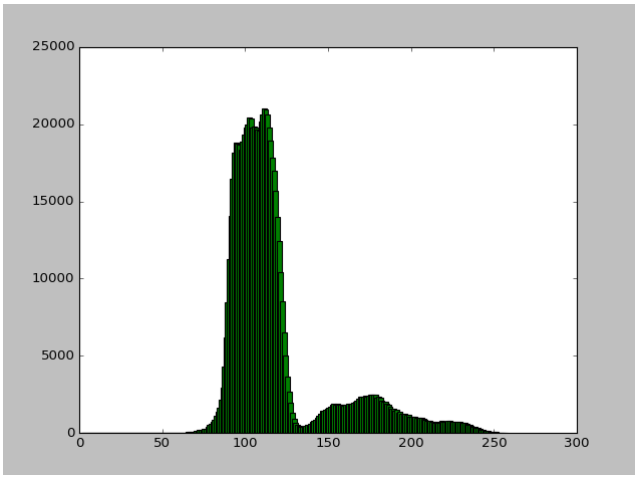
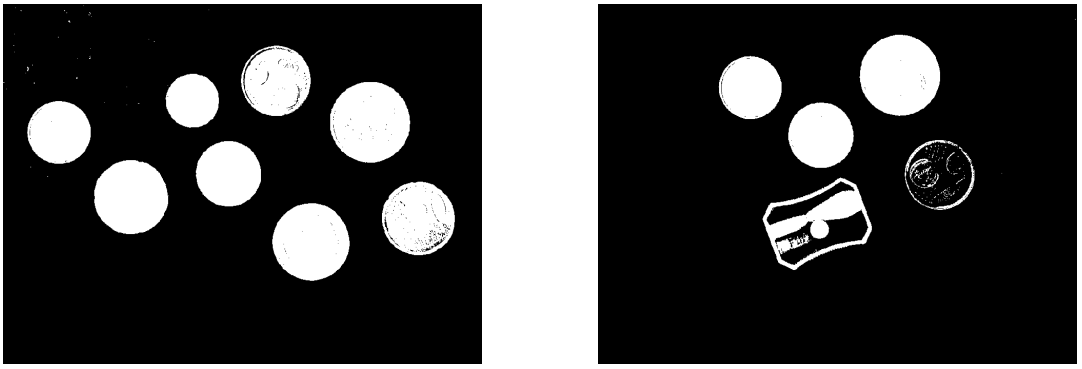


Figure 3.1: *Example of gray-scale image histogram*

the larger of the two describes the background, as it is the dominant part, while the second curve describes the rest of the image that in the specific case is represented by coins and other objects that may be present in the picture.
The goal is to find a value that separates the two regions and to do this the software extracts from the graph three parameters: the maximum, average, and local minimum values. These three values allow you to calculate the value to be used as a threshold in creating the binary

image through the threshold operation and extracting the value is fairly simple as it finds the first local minimum that is after the value maximum of the histogram and less than the average. For example, the histogram of figure 3.1 refers to the input image of figure 2.1a and the resulting binary image after the threshold operation is figure 3.2a, from this histogram the value that the algorithm find is 132 which as you can see is in an excellent point of separation between the two Gaussian curves. However, the binary image is not yet ready to detect the different regions



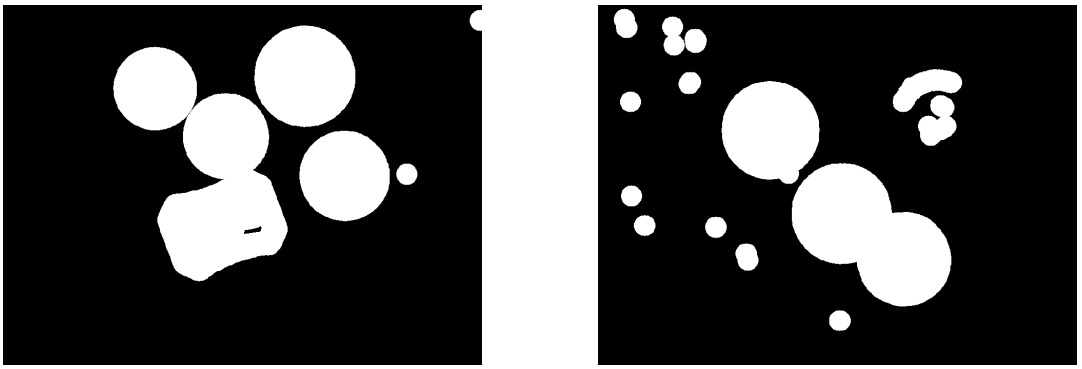
(a) (b)

Figure 3.2: *Examples of binary images obtained with the threshold operation*

representing the objects on the table, as it can be seen in Figure 3.2b, as some coins could be binarized incorrectly and this would lead to the creation of regions not usable. For this reason, some morphological operations must be performed.

3.2 Morphological operations

Morphological transformations are some simple operations based on the image shape. It is normally performed on binary images. It needs two inputs, one is our original image, second one is called structuring element or kernel which decides the nature of operation. Two basic morphological operators are Erosion and Dilation. Then its variant forms like Opening, Closing, Gradient etc also comes into play [1].
In this algorithm, only two basic transformations are used: erosion and dilate.
As seen in figure 3.2b, there is a need to make the homogeneous coin region and to do this, the dilation operation is used with an elliptical structuring element, and an example of the obtained result is visible in figure 3.3a. However, this operation amplifies also all regions that are not of interest (small background points that are beyond the threshold value) and, in the case of figure 3.3b, it makes two coins close to one region. To eliminate these problems, the other basic transformation called erosion applies. The erosion is applied with 4 iterations to have the safety of eliminating anything that is not necessary and to properly separate the regions of interest, an example of result is shown in Figure 3.4a. Finally, in order to give the regions a circular form as circular as possible but still not create any further difficulties for the recognition operation, a dilate operation is performed again but with an different structuring element from the previous one. An example of final result is shown in figure 3.4b.



(a) (b)

Figure 3.3: *Images obtained after the first dilate. Figure (b) represents the dilate of the binary image associated with figure 2.1b*

(a) (b)

Figure 3.4: *Images obtained after the erode operation(a) and after the second dilate(b)*

3.3 Labeling operations and contours recognition

After the individual regions have been isolated, the system recognizes them and the *labeling* function imported from the *bwLabel.py* file is used to do this. This function is equivalent to the *connectedComponentsWithStats* [2] function defined in the OpenCV library and recognizes the different regions within a binary image. With the data obtained from this function, you can create an image that associates each region with a different color, a procedure particularly useful in the case of two neighboring coins to verify that they are actually recognized as separate, as can be seen in figure 3.5.

3.4 Coins recognition

The final part of the algorithm tags the coins according to their value, counts them and calculates the total value in euros in the image.
The first step is to recognize the contours of the regions through the *findcontours* function offered by the openCV library [3]. You can see the contours of these regions with the *drawContours* function and an example of obtained result is shown in figure 3.6.

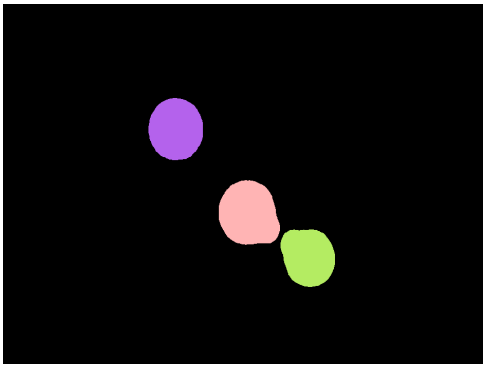


Figure 3.5: *Example of color map image*

After identifying the contours of each region, the coins are labeled by calculating for each of these four parameters: area, radius, perimeter and circularity. After the training images were analyzed, it was seen that each coin had a radius that remains within a certain range of values and this allow to distinguish one coin from another that has a different value.
In figure 3.6 it is possible to note that sometimes some regions are recognized but they are not coins but different objects. In this case, in order to avoid considering them, the value of the circularity is fixed at 15.0, since in the training images the coins always have lower value, while other objects have higher values.
After recognizing the coins, the software have to write over each of them their value and to do so, the moments of each region are extract and you can calculate the coordinates of their centers, then use it as a starting point to write the value.
Finally, with regard to counting the number of coins and the total value in the image, each time the software recognizes a coin, two variables are increased, representing the two interest data and the respective values are then written in the image.

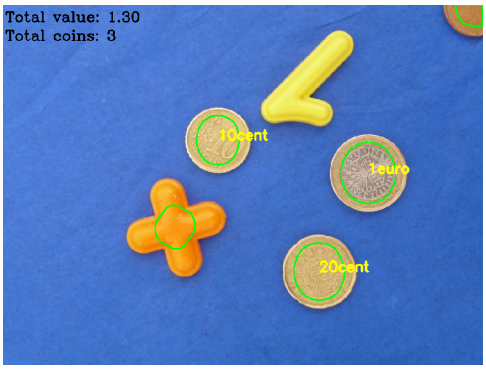


Figure 3.6: *Example of image with region contours*

Chapter 4

Libraries used in the code

If you open the code, you can see how 7 different libraries are imported. Below are some on their features, but in any case, for further details, we invite the reader to use the documentation of each of them.

4.1 OpenCV

This library is imported through import cv2 and is the library that provides most of the features in the code: from morphological transformations to features that allow you to read and view images.
OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. Being a BSD-licensed product, OpenCV makes it easy for businesses to utilize and modify the code.
The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, extract 3D models of objects, produce 3D point clouds from stereo cameras, stitch images together to produce a high resolution image of an entire scene, find similar images from an image database, remove red eyes from images taken using flash, follow eye movements, recognize scenery and establish markers to overlay it with augmented reality, etc. [4]

4.2 NumPy, Scipy and Matplotlib

NumPy is the fundamental package for scientific computing with Python [5]. The use of this library in this software is essential to create the histogram describing the grayscale image and allow to work with this.
However, the library that creates the actual histogram is Matplotlib thanks to the *bar* function and then allows it to be displayed thanks to the *show* function [6].
Finally, in the code you can observe the argrelextrema function that allows you to extract the vector of the local minimum values in the histogram. This function is contained in the SciPy library, a Python-based ecosystem of open-source software for mathematics, science, and engineering [7].

4.3 Math, bwLabel and psColor

The Math library is a Python standard library and it provides access to the mathematical functions defined by the C standard [8]. With this library you can use a series of math operators such as the square root, power elevation, trigonometric functions, etc.
Finally, if you want that the software works properly, *bwLabel* and *psColor* scripts are needed to make labeling and create the so-called "mapColor images".

Chapter 5

Conclusions

If you start the software you can see how the objective that was initially set up, recognize coins and count them, has been achieved, as the application works properly with all the training images.
You can see how the most important steps are 3:

- choose a right value to do a efficient threshold operation. For do this step, it is essential to analyze the histogram that you can extract from the binary image;
- choose the morphological transformations that allow to have "clean" regions;
- find some range of value that allow to label the different coins.

Using the various libraries mentioned in Chapter 4, these steps were taken to reach the set goals, so the software can analyze and extract desired data from all the training images.

Chapter 6

References

1. https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html
2. https://docs.opencv.org/trunk/d3/dc0/group__imgproc__shape.html
3. https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html
4. https://opencv.org
5. http://www.numpy.org
6. https://matplotlib.org
7. https://www.scipy.org
8. https://docs.python.org/2/library/math.html

You can see also:

- "Binary Image Analysis.pdf" , Pedro Mendes Jorge
- Chapter 3 from L. Shapiro, G. Stockman, "Computer Vision", 2001.