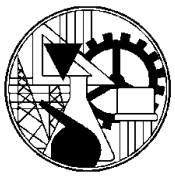


HIGHER EDUCATIONAL INSTITUTE OF ENGINEERING OF LISBON

Computer Science and Multimedia Engineering department

Digital Image Processing

1st Laboratory Project – Coin Detection and Counting

Yeldos Adetbekov

Lisbon, November 2017

ABSTRACT

This project was an attempt at developing an object detection using modern computer vision technology. The algorithm uses adaptive methods to segment the image to identify objects. Incidentally, objects mean euro coins. Main objectives are to develop a computer vision algorithm able to automatically count the amount of money (coins), placed upon a table and familiarization with the OpenCV (Open Source Computer Vision) library to develop real-time computer vision applications (for the Python programming language). In this project work, operations on segmentation of objects are presented stepwisely. The main thesis of all work is thresholding and finding contours algorithms which are included in standard OpenCV library's functionality.

1 INTRODUCTION

In this laboratory project it was developed an algorithm to count the sum of money (euro coins), placed in a table of clear and homogeneous surface and observed by a camera mounted on a tripod, adjusted so that the sensor plane is parallel to the table plane. The algorithm overcomes the following perturbations: (i) the presence of objects other than coins, (ii) presence of minor shadows and (iii) possible contact of objects; To develop the algorithm, training set, which includes different cases of coin placement, has been used.

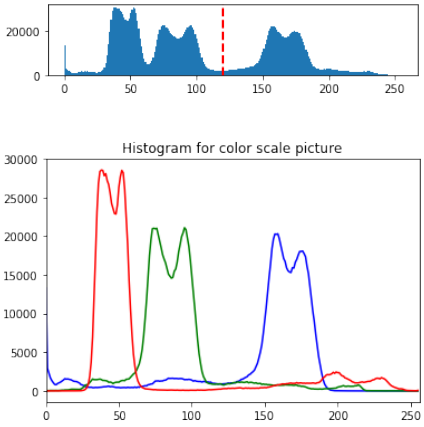
To achieve main objectives, it is obligatorily to evaluate with a test set of images, where algorithm should return stable outputs. This report details the development and implementation of an algorithm that can be used for this purpose. The report first analyses the problem and establishes the requirements for the algorithm. It then examines the major steps of the algorithm and how it meets the requirements. The results of testing the algorithm are discussed before the report concludes with a summary.

2 EXPERIMENTAL AND COMPUTATIONAL DETAILS

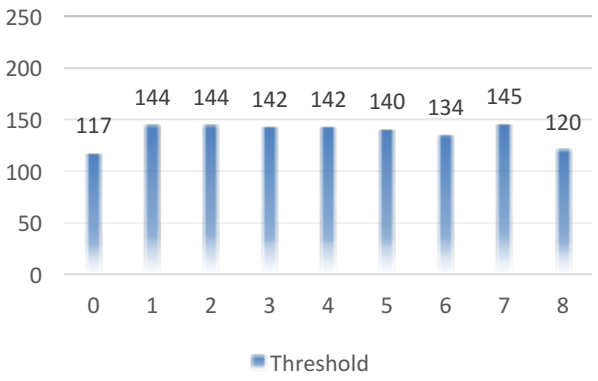
2.1 Thresholding image

Because in every training set's cases the background is homogenous, it is possible to separate background and foreground. Calculating area and shaping are enough to classify coin from other objects. So that, we can use binary thresholding, where 0 (false) is background and 1 (true) is foreground. Binary thresholding algorithm (cv2.THRESH_BINARY) generates the matrix with same resolution (width, height) as initial image, but instead of [0, 255] range it vary in [0,1] range.

To implement thresholding operation, we should calculate optimal value of thresh (minimum value of considerable levels), which is the second parameter of cv2.threshold() function. And for this purposes, we can use adaptive threshold finding algorithm, which called Otsu's method. This method (cv2.THRESH_OTSU) finds optimal value of threshold. In global thresholding, we used an arbitrary value for threshold value. So, how can we know a value we selected is good or not? Answer is, trial and error method. But consider a bimodal image (In simple words, bimodal image is an image whose histogram has two peaks). For that image, we can approximately take a value in the middle of those peaks as threshold value. So in simple words, it automatically calculates a threshold value from image histogram for a bimodal image. (For images which are not bimodal, binarization won't be accurate).

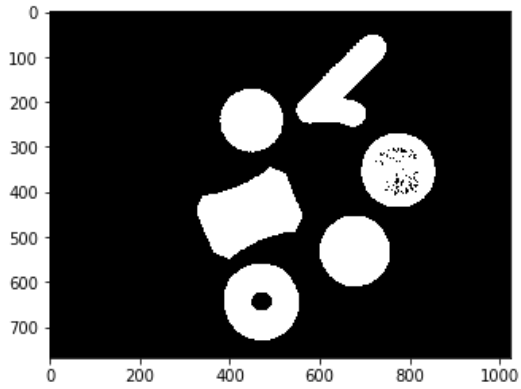


Considering all training set's color histograms, we can say that "RED" channel has high peaks and it can be gathered with "GREEN" channel to one big peak and blue is the background because "R" and "G" distance is less than "B" and "RG" we can consider that they are bimodal. So here we can use adaptive Otsu thresholding algorithm. It is impossible not to say that there is a lot of red in the coins. This decision can be made after a coloring inspection on coins. Here is the statistics on threshold:

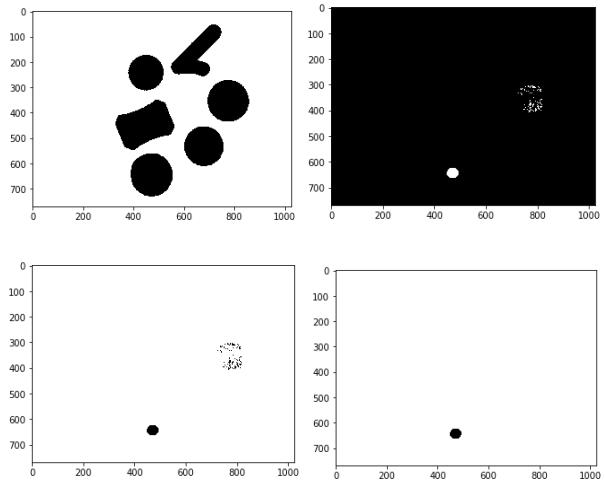


2.2 Filtering noises and masking

After thresholding, we can get new complexities such as noises, which can be gathered like holes inside the coin. That happens because of the structure of coin and the light. We can get this picture:



So that, we need to make the mask, which can overlap all holes, Donut's hole inclusively. For mask we use flood fill functionality of OpenCV, because we say that we have to considering depth in this project, they are background and foreground. We should flood fill from pixel (0, 0), the background to 0 (black), after that just combine inverted floodfilled image with thresholded image. And after that we have simple external first level of hierarchy without any children. But we should to consider hole inside the "Donut", so that we can use morphological transformation trick, what will be considered in the next topic, for getting only donut and cleaning noises:



After that we just can combine considerable hole of donut and mask. We will get cleaned coins from noises.

2.3 Morphological Transformations

Morphology is a broad set of image processing operations that process images based on shapes. Morphological operations apply a structuring element to an input image, creating an output image of the same size. In a morphological operation, the value of each pixel in the output image is based on a comparison of the corresponding pixel in the input image with its neighbors. By choosing the size and shape of the neighborhood, you can construct a morphological operation that is sensitive to specific shapes in the input image.

The most basic morphological operations are dilation and erosion. Dilation adds pixels to the boundaries of objects in an image, while erosion removes pixels on object boundaries. The number of pixels added or removed from the objects in an image depends on the size and shape of the structuring element used to process the image. In the morphological dilation and erosion operations, the state of any given pixel in the output image is determined by applying a rule to the corresponding pixel and its neighbors in the input image. The rule used to process the pixels defines the operation as a dilation or an erosion.

For this concrete objectives (coin detection), we should to know only 2 features, area and shape (circularity). So we know that the shape of coins is circular and to do morphological operation above the image, we should to process it with circular structuring element. For that purposes we can create our own structuring element, such as a disk. It has created by initialization of empty matrix, and using circular formula like $x^2+y^2<=r^2$, we can create validator that will gives us 0 or 1. So here is an example of disk function with parameter 4, where radius is 4, plus 1 as a centroid:

```
[[0 0 0 0 1 0 0 0 0]
 [0 0 1 1 1 1 1 0 0]
 [0 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 0]
 [1 1 1 1 1 1 1 1 1]
 [0 1 1 1 1 1 1 1 0]
 [0 1 1 1 1 1 1 1 0]
 [0 0 1 1 1 1 1 0 0]
 [0 0 0 0 1 0 0 0 0]]
```

I have to say that order what we apply erosion and dilation operations is important. First, used an erode() mathematical transformation to minimize objects countours, due to we have connected (coinsided) elements. Here we have to apply erosion with disk structuring element with radius 8 and make this operation twice, which is can be setted in the iteration attribute. After erosion opperation we have little objects, which are separated from each other. Then, used dilate() mathematical transformation to make it more larger.

2.4 Finding contours

Next step after all filtering, masking, noise reduction and mathematical transformation, we should to classify coins. For that purposes in the standard OpenCV functionality, there is cv2.findContours() function. This function scans our cleaned binary image pixel by pixel from the right-bottom corner to left-top corner. When algorithm get 1, it labels contour by number identifier, which takes digits iteratively. In this stage we can observe the structure (tree) of objects. That is called hierarchy. When the algorithm finds contours, it returns 3 values to 3 variables, where second one is the set of contours and third is the set of contour relation (hierarchy). Hierarchy rows identify [Next object, Previous object, Children, Parent]. In this case we are interested in existence of children and parent. Here to display absence of element, algorithm flag the value as -1 (None).

Donut has a hole. Hole is nested object, in simple words hole object is child of donut object. Respectively donut object is the parent of hole object. This 2 objects have mutually exclusive factors. So that we assign new array of cleaned objects. They represent contours without hole.

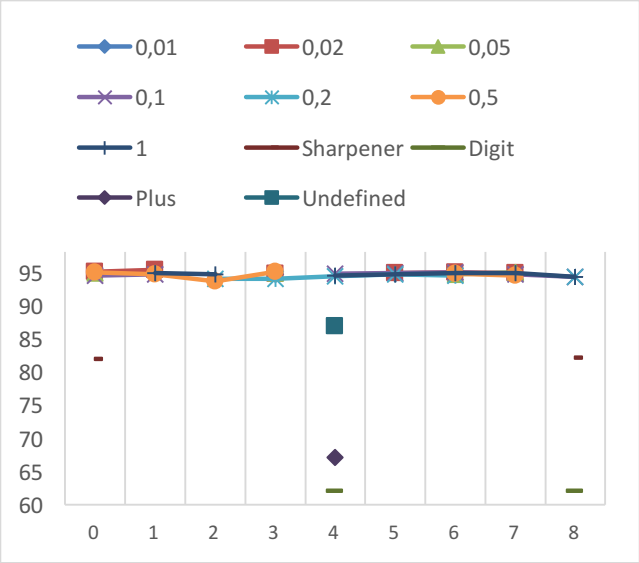
2.5 Validation

Next stage is validating coins from non-coin objects, which are without holes. We remember that there are two main features, which classify objects as coins. They are shape and area. Now for validation, we are interested in the shape of object. Coins are circular and wanted objects are a priori circular. To check the circularity of object there is a mathematical formula:

$$C = \sqrt{\frac{4\pi A}{P^2}}$$

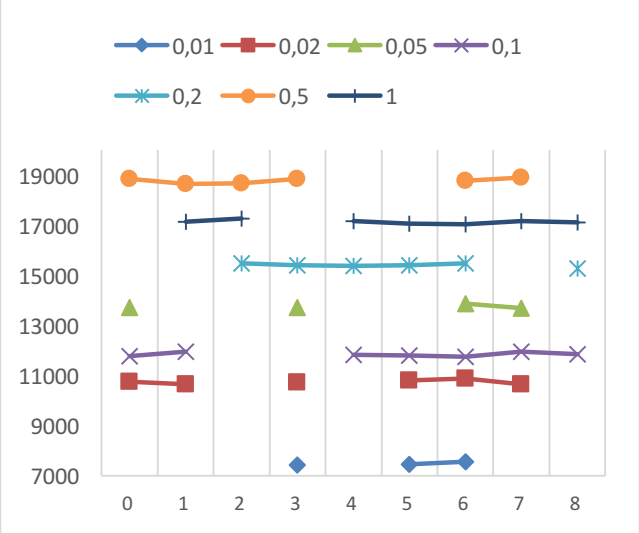
Where A is the area and P is the perimeter (length). This formula calculates percentage of circularity. As higher the percentage is, as the object is similar to circular form. The value varies between 0 and 1 (0%-100%). After estimation we can say that if circularity coefficient is more than 92-93%, it is a circular.

In the following diagram we can see that all of the coins are pretty stable circular. Quarter of coin on the right-top of test image number 4 is more circular, it is about 86-87% of circularity. That is because it has circular surface, but not full circular. Next one is sharpener, it is about 81-82% of circularity. That is because this object has no a lot of sharp corners (angles). Next is plus sign, it is pretty round, but has 4 cavities. The circularity is about 67%. And digit has the lowest percentage of circularity, because it is not proportional (one side is more than second) and it has sharp angles.



2.6 Counting classified coins

This part is more easy, we just should to match areas and coin denominations. Here we assign two arrays, one is coin denominations and second is coin areas, which correspond to the coin values.

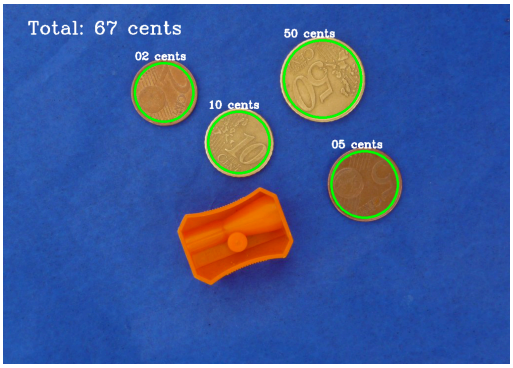


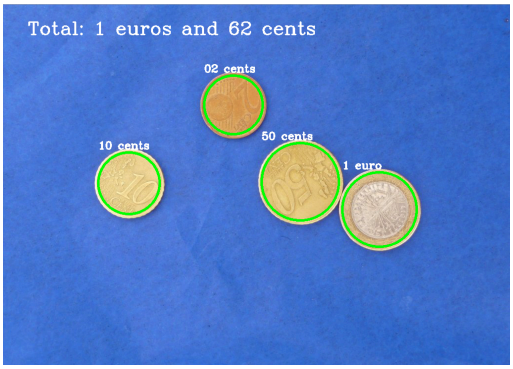
Here is the table of ranges:

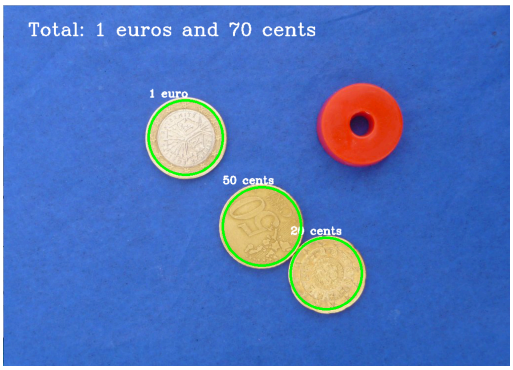
	0.01	0.02	0.1	0.05	0.2	1	0.5
Min	7429	10646	11737	13689	15264	17030	18651
Avg	7475	10740	11841	13751	15402	17139	18782
Max	7549	10876	11957	13874	15482	17281	18901

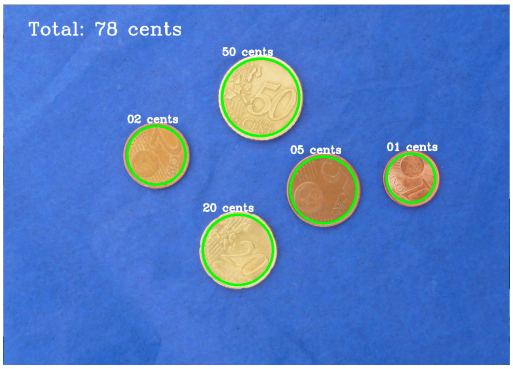
3 RESULTS AND DISCUSSION

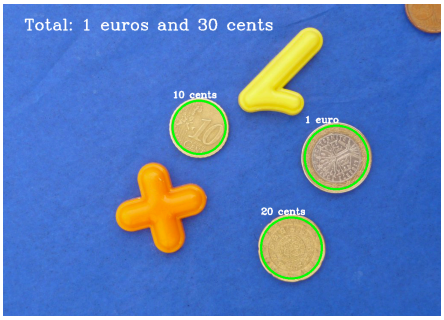
As a result we summarize values of all classified coins and display the result in the left-top corner by putText() function.

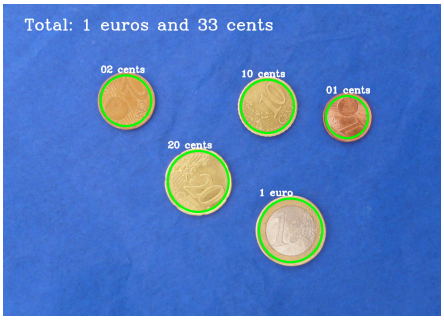


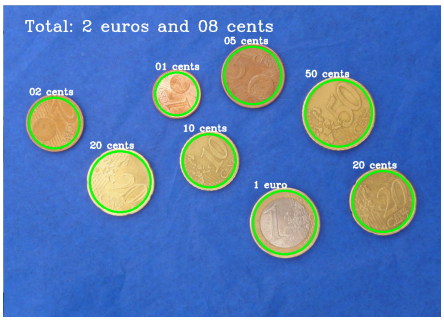


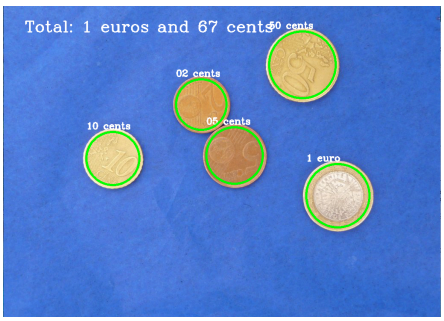


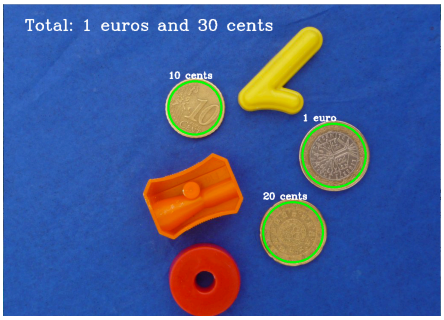












4 CONCLUSIONS

In summary, we have performed both an experimental and theoretical study of the thresholding, masking, filtering and finding contours. The experimental results have been successfully interpreted by listed operations. Main problems (connected coins, noises inside coins, objects with the hole, objects that are not circle, size) is done as estimated.

This algorithm could be extended and finalized by another additional instruments, such as watershed and pattern recognition and predication (skimage).

We got some much needed practical experience with image processing and were able to learn about multiple techniques and strategies.

A HEADINGS IN APPENDICES

A.1 Introduction

A.2 Experimental and Computational Details

A.2.1 Thresholding image

A.2.2 Filtering noises and masking

A.2.3 Morphological Transformations

A.2.4 Finding contours

A.2.5 Validation

A.2.6 Counting classified coins

A.3 Results and Discussion

A.4 Conclusions

A.5 References

ACKNOWLEDGMENTS

This work was supported by assistant professor Pedro Mendes Jorge.

REFERENCES

[1] L. Shapiro, G. Stockman, “Computer Vision”, 2001, Prentice Hall, ISBN – 0-13-030796-3;
[2] R. Gonzalez, R. Woods, “Digital Image Processing”, Pearson International Edition, 3rd edition, 2008, ISBN: 0-13- 505267-X;
[3] Mubarak Shah, Fundamentals of Computer Vision, December, 1997, on-line publication;
[4] Joseph Howse, OpenCV Computer Vision with Python, PACKT Publishing, 2013;
[5] D. Baggio et al, Mastering OpenCV with Practical Computer Vision Projects, PACKT Publishing, 2012;