FACOLTA’ DI INGEGNERIA

Computer Vision Course

Final Project

**Coin Detection And Classification**

|  |  |
| --- | --- |
| By:  Sartori Leonardo  Matricola 1207006 | Prof. Ghidoni |

Academic Year 2018-2019

Index

|  |  |
| --- | --- |
| * Introduction | 3 |
| * Pipeline overview | 5 |
| * Circle detection |  |
| * Coin recogniction |  |
| * Performance analysis |  |
| * Conclusion |  |
|  |  |
| * Bibliography | 35 |

1. Introduction

The purpose of this project is to analyze an input image in order to detect if one or more possible coins are present and then classify these candidates in order to establish if each of them is an Euro coin. The tools that I used in order to achieve that are several and from various source and programming language. In fact, this is not an easy task, as it is necessary to firstly include several possible circle candidates, allowing some false positives but not a huge amount of them, and then to rely on the ability of a neural network to establish if a candidate is an Euro coin, and which one of the 8 coin possibilities it represents, or not.

The starting point is the circle detection, a typical task in Computer Vision field that we already faced up in one of the laboratory experience. As in that case, I used C++ language and OpenCV library in order to be able to implement some common techniques in order to extract circles and get some sub-images of the input one, which include only one circle.

Starting from these images, a Python script runs a neural network in order to classify each of them in one of the Euro coins classes or in an Unknown class. Finally, the script saves the results of the classification and the core C++ script reports the results in ordered way.

A model that can sum coins and tell how much money we have in a group of coins could be used for people with vision disabilities. Can Computer Vision and Deep Learning effectively enable to count, classify and sum in a single pipeline? What can be done with a relatively small amount of data? Can we achieve a good generalization and predict sums beyond the data-set images? I tried to do that and achieve some good results. Obviously the problem is quite complex and some advanced techniques should be implemented in order to achieve completely this goal.

2. Pipeline overview

The pipeline of my project is quite simple but needs some explanations. Firstly, the C++ program named *main.cpp* is the core and the coordinator of the all process. This parses user input, retrieves the selected image from the file system and acts some OpenCV preprocessing in order to find good candidates to be coins. This is done through a custom class *Recognizer.cpp* that does the circle detection job. After that, the scripts saves these candidates as new and smaller images in the file system and calls a Python script named *test.py* to run a neural network classifier on them in order to a get a prediction for each candidate. When this process ends, *test.py* produces a *.csv* in order to store the results and made them easily readable by the core *main.cpp*. In the end, this last, parses the *.csv* into a string vector and counts how many predictions of each class were recognized, printing them and their sum in Euro.

3. Circle detection

I started from the previously mentioned activity in the course laboratory regarding street signs detection and tried to transfer my experience on coins images. I realized soon that simply applying HoughCircle, an OpenCV function that detects circles in an image using Hough Transform, was not enough to determine good circles in different images.

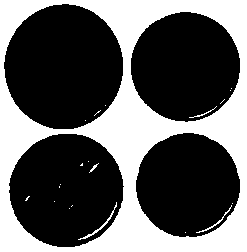
So I started wondering about the some kind of image preprocessing necessary to improve circle detection. I read several blogs, guides and articles about that and started trying some basic image transformation in order to clarify circles present in the image, before feeding them to HoughCircle. Starting from an article on a blog [1] and manipulating the process presented in order to adapt it to my case, I built a preprocessing pipeline through which I was able to extract good circle candidates from all the provided images.

  
Example of original image (1.jpg)

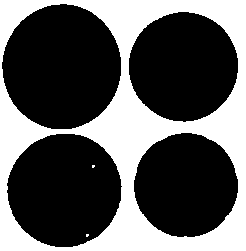
Firstly I convert the original image into gray, as the following method can only manage this type of images. Then I do a Gaussian Blur with a kernel of size 15x15, in order to remove noise in the image, smooth all the lines I will use as edges and reduce false edges.

Example of blurred image

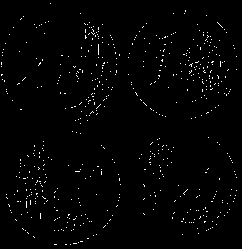
Then I though it will be useful to threshold the image to obtain only a binary image, in which a color is used for the background and the other is used for the coins. Regarding this, I started with adaptive thresholding method but the results were not so good and I was not obtaining the prevented image. So I decided to implement Otsu thresholding, which uses the histogram of the image to calculates a threshold to divides the pixel color values into two groups, below and over the threshold, and colors the image with black and white consecutively.

Example of thresholding

At this point, I had

Example of closing

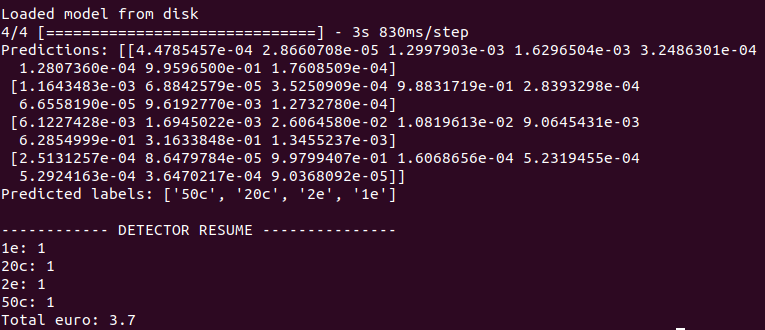
At this point, I had

Example of Canny result

At this point, I had

Example of final circle detection

At this point, I had



Bibliography

1. http://blog.christianperone.com/2014/06/simple-and-effective-coin-segmentation-using-python-and-opencv/
2. Chris M. Bishop,