University of Padova

ICT for Internet and multimedia

Computer Vision

Coin Detection & Classification

Student Name:

Qamar F. Alfalouji

1203278

Professor:

Prof. Ghidoni

❖ Introduction:

Coin detection and counting is a very important need of human being and as the life depends solely on machines, the detection can be done using several techniques and approaches, and this report discusses how to solve this problem in two main steps:

1) Coins recognition and segmentation using Hough transform (OpenCV).
2) Coins recognition and classification using machine learning.

Fig (1): the main used algorithm steps

The performance rate of detection and recognition was more than 95% as computed by Neural networks.

Step 1: Coin detection and segmentations:

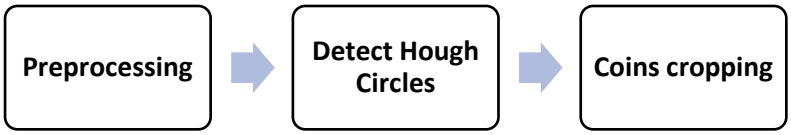

Fig (2): the main steps for detection and segmentation.

- All the experiments are done to the images available in T2 folder in elearning
- A class for coins detector named CoinDetector is defined which offers functions that implement the detection in several steps, as follows:

1) Preprocessing (CoinDetector.preprocessing(cv::Mat *input_img))
   1. Convert the image to grey using the function `cvtColor with color CV_BGR2GRAY, in order to process it later.`
   2. Reduce the noise by using Gaussian blur function `GaussianBlur.`

A B

Fig (3): A: the original photo before the preprocessing, B: The result image after the preprocessing

2) Detect Hough circles: CoinDetector.computeHoughCircles(cv::Mat src)

In order to detect the circles in the image, there are many different approaches that can be used, the one used in this project is detecting the Hough circles using the function offered by opencv:

HoughCircles(InputArray **image**, OutputArray **circles**, int **method**, double **dp**, double **minDist**, double **param1**=100, double **param2**=100, int **minRadius**=0, int **maxRadius**=0)

With the following arguments:

- **minDist:** defines the minimum distance between the centers of the two detected circles, and after many trials**.**
- **method**: set to be CV_HOUGH_GRADIENT.
- param1 – First method-specific parameter. it is the higher threshold of the two passed to the Canny() edge detector (the lower one is twice smaller).
- param2 – Second method-specific parameter. it is the accumulator threshold for the circle centers at the detection stage. The smaller it is, the more false circles may be detected.
- minRadius – Minimum circle radius.
- maxRadius – Maximum circle radius.

Configuring the previous function can yields to too many possible values for the arguments, and in order to produce something general and can be implemented to all the dataset, the following facts are considered:

- the highest ratio of the dimensions between the coins is between the 2 euros(the largest with 25.75 mm) and 1 cent (the smallest with 16.25 mm) so the maxRadius will be the mentioned ratio multiplied by the minRadius.
- The minRadius value can be different so we can just start by a value and loop for a given number of iterations and compute the Hough circles in each iteration, the settings that gives us the largest number of the circles is supposed to be the good ones.
- The canny threshold can be computed in several ways, one of them is to compute the mean of the whole image.

A B

Fig (4): A: the original photo before detection, B: The result image after detection

A B

Fig (5): A: the original photo before detection, B: The result image after detection

3) Coins cropping and saving:

After detecting the circles, the function of CoinDetector.cropAndSave() crops all detected circles and save each one as stand-alone circle with new name.

Step 2: Coins classification using machine learning (Keras, tensorflow)

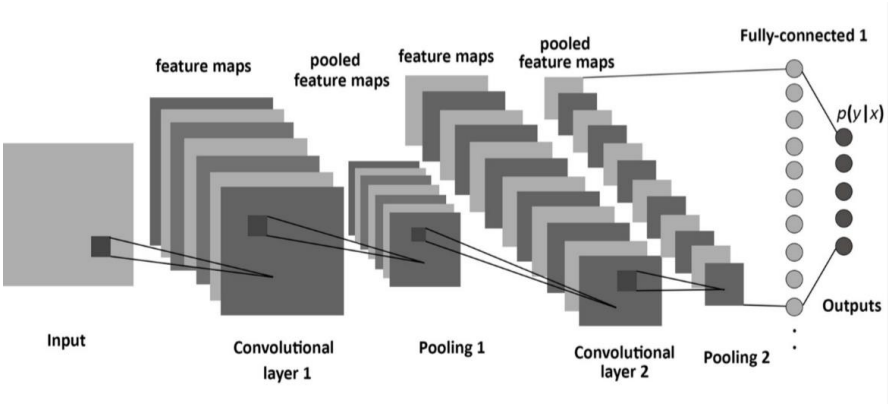

Fig (5): The general structure for the convolutional network.

The recognition and classification of the detected coins can be done using machine learning by:

1. Constructing a CNN model.
2. Training the model using the dataset available here:https://github.com/kaa/coins-dataset
3. Testing the model on the cropped images.

The most accurate model achieved has the following structure:

- 4 convolutional layers using the function of `Conv2D with kernel size 3x3 and filters as 16,32,64 and 128.`
- 4 max pooling layers using the function MaxPooling2D
- 1 dense layer.
- 1 flatten layer.

The following description can be obtained using model.summery() function provided by keras:

conv2d_1 (Conv2D)	**(None, 150, 150, 16)**	**448**
max_pooling2d_1 (MaxPooling2	**(None, 75, 75, 16)**	**0**
conv2d_2 (Conv2D)	**(None, 73, 73, 32)**	**4640**
max_pooling2d_2 (MaxPooling2	**(None, 36, 36, 32)**	**0**
conv2d_3 (Conv2D)	**(None, 34, 34, 64)**	**18496**
max_pooling2d_3 (MaxPooling2	**(None, 17, 17, 64)**	**0**
conv2d_4 (Conv2D)	**(None, 15, 15, 128)**	**73856**
max_pooling2d_4 (MaxPooling2	**(None, 7, 7, 128)**	**0**
dense_1 (Dense)	**(None, 7, 7, 128)**	**16512**
flatten_1 (Flatten)	**(None, 6272)**	**0**
dense_2 (Dense)	**(None, 8)**	**50184**

Fig (7):The model architecture.

The mentioned model gave an accuracy percentage that is more than 95% and the result can be observed as it will classify the cropped images into the folders of classes of the coins and it will write the class name on each image, the classes are:

'10c' : 10 cents, **'1c'**: 1 cent, **'1e'**: 1 euro,**'20c'**: 20 cents, **'2c'**: 2 cents, **'2e'**: 2 euros, **'50c'**: 50 cents **'5c'**: 5 cents.

And here are some of the results:

Fig (8):The content of the folder '2e' after running the program on some images from the dataset.

A

B

Fig (9): A: Two euros coin detected as '2e', B: One euro coin detected as '1e'

As noticed in figure 9, the coins are labeled with the class name from the provided class names that were mentioned previously.

Cofusion matrix:

```
[[ 55   9  76  43   3   6  42 124]
 [  9   3   7   7   0   0   3  17]
 [ 35   7  50  24   1   4  20  85]
 [ 42  10  56  54   3   4  33 121]
 [  1   0   3   2   0   0   0   9]
 [ 14   1  11   6   1   0   4  14]
 [ 15   2  24  19   0   4  13  35]
 [ 49  13  65  56  11   5  40 171]]
```

The mentioned performance and structure came after many trials, by changing the number of layers, changing the number of epochs, steps be epoch and many other parameters, and the provided one was the best one, and I provided 2 other models which can provide some different results for some images. However, the 100% performance is difficult to be achieved and there is always some percentage of errors, such as predicting that the 1 cent is 5 cents as shown below:

Fig (10): Wrong classification for 1 cent as '5c'.

❖ References:

1) https://pdfs.semanticscholar.org/c062/331dd08338ae8f31d54065d654eee562c01b.pdf
2) https://pdfs.semanticscholar.org/316d/e49a4af48a8fb3d75719c41eb6dba72ac452.pdf
3) **'Implementing Deep Networks in Keras' by prof.Stefano Ghidoni, 2018-2019**