Project 9
Coin Recognition

Christoph Fuchs (0625267, 066 932)
Roman Hochstöger (0627154, 066 932)
Martin Schacherl (0426435, 066 939)
Vienna University of Technology

Abstract

In this report, a methodology for detecting coins in images is shown. Our approach is based on Hough transformation for segmentation, and Log-polar transformation in combination with Fourier transformation for getting features. For classification, a support vector machine and a neural network, using the euclidian distance are used. Moreover, our results, limitations, and improvement of our approach are discussed.

1 Introduction

Coin recognition is a non-trivial problem to solve. The goal of coin recognition is to detect coins in an image and register the corresponding values of the coins. There are several difficulties regarding coin recognition, which include

- different perspectives of coin images
- coins which are partially overlapped (by some other coins or other objects)
- rotation angle of the coin
- frontside and backside of the coin show different embossings
- regarding Euro coins, backside embossings vary by country
- light reflections
- impurities

Much has been written about coin recognition in the past. The authors of [Che10] presented an approach using Log-polar transformation and fourier transformation. In [Kam08] a solution using the Scale-invariant feature transform (SIFT) algorithm. Moreover, [ea00] used the genetic algorithm and simulated annealing for this purpose.

In this paper, we will give an overview about the algorithm we used in Section 2. After that, we provide some information about how we tested our algorithm in Section 3. Finally, we will disuss the results in Section 4 and draw our conclusions in Section 5.

2 Methodology

The algorithm presented follows the approach stated in [Che10].

2.1 Segmentation

Segmentation is done by Hough transformation. Some preprocessing is necessary to use the Hough transformation: The image is first transformed into greyscale color space, followed by edge detection using the canny edge detection algorithm, which leads to a binary image, where the edges of coins (and other objects) are represented by white 1 pixel wide lines on a black background.

To reduce complexity and therefore increase speed, we only search for circles with a defined minimum/maximum radius in the binary image. This leads to some limitations concerning the perspective the coin image is taken. We will discuss that in detail in Section 4.

The segmentation process is done by transforming the binary image into the Hough space. To achieve this, a circle of given radius is calculated for every white pixel, using this pixel as center point, increasing the grey value at the circle edge by 1.

After that, accumulation points - pixels with maximum grey value - are collected and assumed as center of coins in the image.

In figure 1 a visualisation of the Hough space is shown, in figure 2, we can see the results of segmentation.

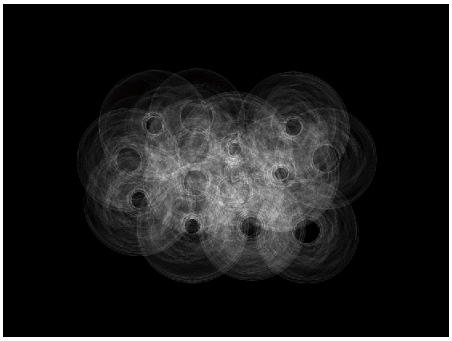

Figure 1: Visualisation of Hough space. Some accumulation points can be seen

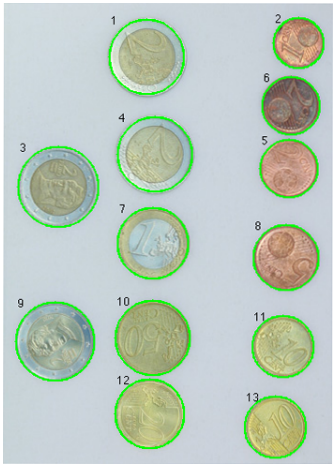

Figure 2: Result of segmentation process

2.2 Feature extraction

In order to extract the desired features, some more preprocessing steps are necessary. To facilitate rotation invariance, the pixel coordinates are transformed to polar coordinates, using the log-polar transformation

$$r = \log_a \sqrt{(x - x_c)^2 + (y - y_c)^2} \quad (1)$$

$$\theta = \arctan\left(\frac{y - y_c}{x - x_c}\right) \quad (2)$$

Rotation in the spatial domain is equal to x-axis shift in the log polar domain, which enables us, to cope with different rotation angles easily. The results of log-polar-transformation using a 2 euro coin-image are shown in figure 3

Figure 3: Log-polar-transformation of a 2-euro-coin image

After that, the transformed image data is row-wise transformed to the Fourier space. The algorithm calculates 50 Fourier coefficients per line, but only coefficients 2-16 are used. The first coefficient is not used, because it represents the mean grey value of the image, and therefore contains no information of interest in order to solve the coin recognition problem. We decided not to

use coefficients 17-50, because the significance of information contained by these coefficients is decreased, and moreover, we desired to achive better time performance for our algorithm.

The Fourier-transformed image is divided in ten 10% segments along the y-axis. This is done to achive better size-invariance of our algorithm: Due to the fact that the image data was transformed to the log-polar space before, each segment represents a concentric cirle around the coin's center. A visualisation of such circles can be found in figure 4.

Figure 4: Concentric cirles, generated by y-axis segmentation

2.3 Classification

The features extracted from the images are mean value and variance. For classification, we tried both a neural network using the euclidian distance, and a polynomial Support Vector Machine. The results will be discussed in the next sections.

3 Experiment

We have made a database consisting of austrian Euro coin images, with 20 coins per image. For taking the photos, a selfmade tripod was used. The experiment was done using a Support Vector Machine, and a neural network using the euclidean distance. The test setup consists of two tests – one to detect if either the front- or the backside of the coin is visible, the other to detect the value of the coin. We also tried some improvements, which will be discussed in section 4. The test results are shown in table 1. Some trainingset-images can be found in figure 5

4 Disscusion

4.1 Interpretation & Improvement

The results above show very promising results. In comparision with results found in the literature, our algorithm performs quite good. There is no denying we tested our algorithm under optimal condidtions. Limitations of the algorithm will be discussed in section 4.2. An result example image is shown in figure 6.

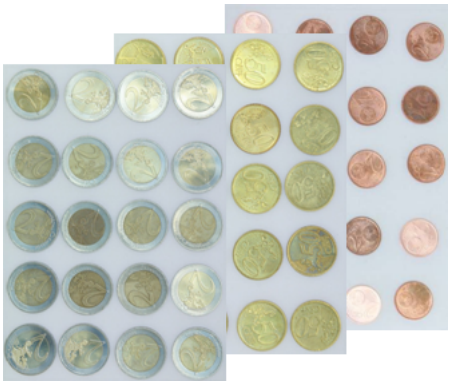

Figure 5: Some images of our training set

SVM amount	SVM coin side	EUK amount	EUK coin side	
0.8165	0.6804	0.7753	0.3924	Greyscale
0.9082	0.7057	0.8924	0.4304	Greyscale + color channels
0.9177	0.6994	0.8892	0.4241	Color channels
0.9177	0.6994	0.8892	0.4241	Color channels + radius

Table 1: Results of the coin recognition algorithm

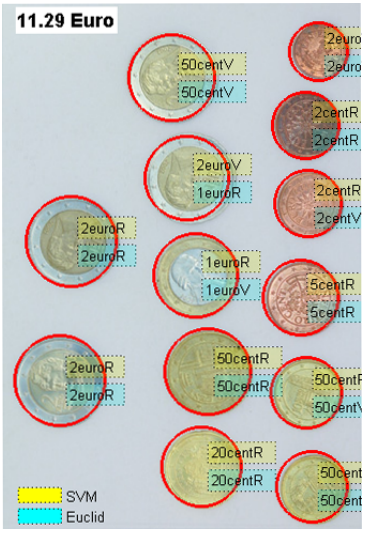

Figure 6: Result of classification. Some classification errors can be seen.

Moreover we enhanced the algorithm's performance by adding an improvement: As shown in table 1, doing the classification color channel-wise yields to much better results than classifying the greyscale images. We also found out, that especially the blue color channel contains much information, presumeably because the edges appear strong in this channel. We also tried to improve algorithm's performance by using the coin radius as a feature, surprisingly this effort had no influence on the performance, as can be seen in the last line of table 1.

4.2 Limitations

There are also some limitations of our algorithm. In the segmentation process, only coins with circular shape are detected, in other words, only images taken perpendicular to the coins are processed. The reason is, that detecting elliptical shapes would lead to heavy time performance issues, because ellipses are described with more than one parameter.

Moreover, the Hough transform lacks in segmenting occluded coins. In this case, not all coins are detected. Only some accumulation points are generated by the Hough transform, but it is not possible to distinguish between true and false positives.

In addition, there are some restrictions concerning the image parameters. The image must be taken at a constant distance (which we achived by our selfmade tripod), the background must be plain-colored, and illumination must be bright and diffuse – there must not be heavy light reflexions or coin shades.

A limitation of our *MATLAB* code implementation is, that the code only works with a 32bit version of *MATLAB*.

5 Conclusion

This paper introduced an algorithm to detect coins in images and calculate their values. The test setup and our results are discussed. The algorithm follows mainly the work in [Che10]. Moreover, some improvements to this algorithm and its limitations are presented.

References

[Che10] Cai-ming et al. Chen. A coin recognition system with rotation invariance. In *Proceedings of the 2010 IEEE International Conference on Machine Vision and Human-machine interface*, pages 755–757, 2010.

[ea00] Mitsukura Yasue et al. Design and evaluation of neural networks for coin recognition by using ga and sa. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks*, pages 178–183, 2000.

[Kam08] Zaharieva M. Kampel, M. Recognizing ancient coins based on local features. In *Proceedings of the 4th international symposium on visual computing*, pages 11–22, 2008.