A Fast and Reliable Coin Recognition System

Marco Reisert, Olaf Ronneberger, and Hans Burkhardt

University of Freiburg, Computer Science Department,
79110 Freiburg i.Br., Germany
reisert@informatik.uni-freiburg.de

Abstract. This paper presents a reliable coin recognition system that is based on a registration approach. To optimally align two coins we search for a rotation in order to reach a maximal number of colinear gradient vectors. The gradient magnitude is completely neglected. After a quantization of the gradient directions the computation of the induced similarity measure can be done efficiently in the Fourier domain. The classification is realized with a simple nearest neighbor classification scheme followed by several rejection criteria to meet the demand of a low false positive rate.

1 Introduction

The goal of a coin recognition system is to automatically sort and classify high volumes of coins with high accuracy within a small amount of time. In 2003 ARC Seibersdorf research GmbH created the sorting device called *Dagobert* [1]. The recognition unit of Dagobert is able to discriminate between over 600 different coin types based on over 2000 different coin faces. In 2006 ARC Seibersdorf formulated together with the MUSCLE[1] Network of Excellence and the PRIP, Vienna University of Technology the *Coin Image Seibersdorf (CIS) Competition 2006*[2] to foster the development of robust coin recognition algorithms. The present paper proposes the price winning algorithm.

The proposed system roughly follows the ideas in [1]. The similarity of two coin images is computed by the use of registration techniques. In a first step the translational pose of the coin is determined by a segmentation algorithm that makes an estimate of the coin's radius and its center. The comparison of two coins is done by aligning them with respect to their rotational pose, i.e. we have to optimize only one parameter, which makes the registration feasible. Having defined a similarity measure any classification scheme may be used. Because of the highly reliable embossing process for the coins we believe that a registration technique is the first choice to reach good results. The only difficulty is to find robust similarity measures that tolerate the, sometimes severe, abrasion and fouling of the coins, but still give response for the reliable embossment which determines the class membership. Additionally this similarity measure has to be

[1] http://www.muscle-noe.org/
[2] http://muscle.prip.tuwien.ac.at/index.php

F.A. Hamprecht, C. Schnörr, and B. Jähne (Eds.): DAGM 2007, LNCS 4713, pp. 415–424, 2007.

computable in a fast manner such that the resulting algorithm can cope with large databases.

The article is organized as follows: In the following subsection we give a short overview over related work. In Section 2 we present our algorithm for coin segmentation that is based on the Hough transform. In Section 3 we present the features that are used for alignment and similarity computation followed by some implementation details. The classification scheme is presented in Section 4. Finally we show results on the CIS Benchmark datasets. In Section 6 we give a conclusion and ideas for further improvement.

1.1 Previous Work

Recent approaches for coin recognition can roughly be divided in methods based on rotational invariant features and methods based on registration. In [5,6] invariant features are used to compare coins in a rotational invariant manner. In [5] very high false positive rates are reported. In [1,7] the similarity computation is based on registration techniques. It seems possible that these approaches are able to fulfill the reliability demanded in the benchmark specifications.

2 Coin Segmentation

In [1] a simple segmentation scheme is used which works by thresholding the grayvalues. This approach is based on the assumption that the coins itself are brighter than its background. Having a look at the benchmark database this is mostly but not always the case. So, we have to search for another solution. The Hough transform is known to be a very robust segmentation tool. We use a generalized Hough transform (GHT) [2] to segment the coins. In [3] the same method was used for a fast segmentation of cell-nuclei. We use a three dimensional voting space, namely the two coordinates of the coin's center and its radius r. Let us call the gray-valued image function $I : \mathbb{R}^2 \mapsto \mathbb{R}$ and the voting function $P : \mathbb{R}^3 \mapsto \mathbb{R}$. The idea is to let each pixel cast a vote for possible circle centers at particular radii. To keep the running time low the votes are performed only for those points that are colinear with current gradient vector. Formally we search maxima of the following function

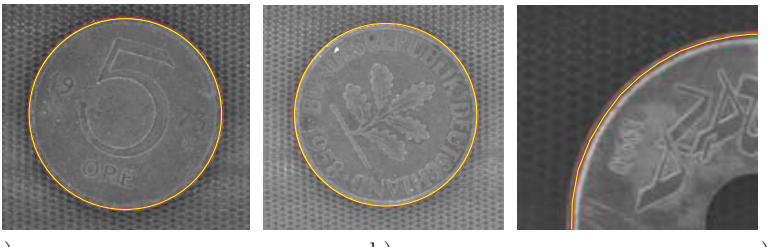
$$P(\mathbf{x}, r) = \int_{\mathbb{R}^2} \left(\delta\left(\mathbf{x} + r \frac{\nabla I(\mathbf{y})}{||\nabla I(\mathbf{y})||} \right) + \delta\left(\mathbf{x} - r \frac{\nabla I(\mathbf{y})}{||\nabla I(\mathbf{y})||} \right) \right) \; ||\nabla I(\mathbf{y})|| \; d\mathbf{y},$$

where δ is some indicator function giving contribution whenever its argument is nearby zero, for example a gaussian or a rectangle-function. Each gradient in the image votes for a possible center of the coin, where the hypothetical center has to be along the gradient's direction. Since we do not know whether the coin is darker or brighter than the background we have to vote in positive and negative direction.

Before starting to compute the integral we blur the image with a first-order IIR-Filter to get smooth gradients. The computation of the integral is straightforward. Just run linearly over the image I and compute the gradient $\nabla I(\mathbf{y})$ and

its magnitude. Now for every position $\mathbf{y}$ and for discrete radii values r_i accumulate the voting function at positions $\mathbf{y} + r_i \frac{\nabla I(\mathbf{y})}{||\nabla I(\mathbf{y})||}$ and $\mathbf{y} - r_i \frac{\nabla I(\mathbf{y})}{||\nabla I(\mathbf{y})||}$ with weight $||\nabla I(\mathbf{y})||$. The shape of the accumulation depends on the indicator function δ. We just round the estimated coordinates to the nearest integers and accumulate the single pixels. After all accumulations are done the voting function P is smoothed by an IIR-Filter. Hence, the indicator function is just the impulse response of this filter.

Since we know that there is only one coin present in a image, we can use a hierarchical voting scheme to get better estimates for the radius. For a first rough estimate we take 16 different values for the radius covering the whole range of possible coin sizes. After the determination of the first maximum we distribute again 16 radius bins around the first rough estimate. Overall we repeat this procedure four times resulting in an accurate estimate. On an *Intel P4 2.8Ghz* the overall segmentation procedure needs less than one second for one image. In Figure 1 we show three examples of segmented coins.



a) b) c)

Fig. 1. Coins segmented by our algorithm. Coin a) and b) have very bad contrast conditions, however our algorithm is able to make a good segmentation. In c) a section of a coin is shown where our algorithm makes a small mistake. Due to a strong circle-like structure at the border of the coin the estimated radius is a little bit too small. However, this is a systematical error, and if it happens for every coin from the specific class it shall not confuse the classification algorithm.

3 Feature Extraction and Registration

Having a look at the benchmark database one can guess that the actual gray values of the images are not very discriminative. In [1] a Canny Edge-Detector is used to compute more reliable features. This is, of course, a much better idea than working on the plain gray values, but it has still some disadvantages. At first, the results depend heavily on the choice of the parameters of the edge-detector. These parameters have to be chosen properly depending on the illumination conditions and the quality of the images. And secondly, the orientation of the edges are completely neglected. We want to follow a different approach. We want to use solely the direction of the gradients in the image and totally neglect its magnitude. This has the advantage that we are independent of illumination and

contrast changes. One can argue that only considering the direction is a very dirty approach, since we also compute gradient directions in homogeneous, flat regions where theoretically the gradient has to be zero and hence no reliable direction exists. But however, there are several reason to follow this idea. First, if the direction in homogeneous regions can be assumed to be equidistributed with respect to its angle, then the similarity measure can be designed such that this regions give only a constant bias. Further, we do not need any threshold, i.e. we never need to decide whether there is an edge or not. This advantage becomes important considering Figure 2. Due to abrasion coins show typical patterns near structural steps and edges. One can see several slight gradients near the edges. It would be ignorant to neglect this information. An algorithm, which makes only use of edges has problems to also incorporate this information.

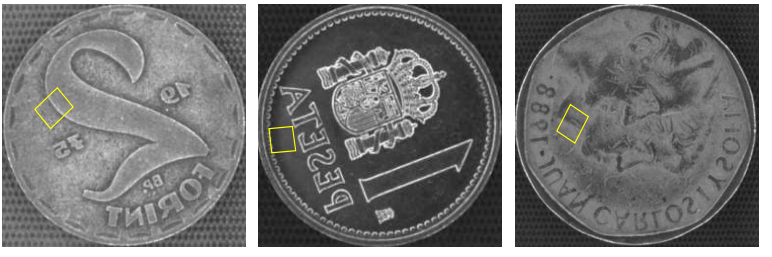


Fig. 2. Abrasion effects. Due to abrasion several slight gradients are introduced near the edges. Not only the edges contain structural information that may be considered, also putative flat regions contain valuable structure. Our algorithm also uses this information for recognition.

During segmentation we determined the radius and center of the coin. From now on we assume that the center is shifted to the origin and the radius is normalized to 1. For convenience we represent the image I in polar coordinates $I(r, \varphi)$ with $r \in [0, 1]$ and $\varphi \in [0, 2\pi]$. The basis for our features is the normalized gradient image $\mathbf{g} = (g_r, g_\varphi)^T$ given by

$$g_r(r,\varphi) = \frac{\partial_r I}{\epsilon + \sqrt{(\partial_r I)^2 + (r\partial_\varphi I)^2}}, \quad g_\varphi(r,\varphi) = \frac{r\partial_\varphi I}{\epsilon + \sqrt{(\partial_r I)^2 + (r\partial_\varphi I)^2}},$$

where ϵ is a small positive constant avoiding division by zero. We choose the radial and tangential derivatives because they do not change while rotating the coin. A rotation of the coin just shifts the φ coordinate of $\mathbf{g}$ cyclically. Based on the gradient image we compute the angle image $a \in [0, \pi]$ given by

$$a(r,\varphi) = \operatorname{sign}(g_\varphi(r,\varphi)) \arcsin(g_r(r,\varphi))$$

describing the angle to the tangent of the circle. The sign modification leads to invariance against inverting the contrast. We found that it is not important whether an edge is descending or ascending, because, e.g. by dirt or abrasion of

the coin, the contrast conditions are sometimes inverted. So only one half of the circle is represented by this function, the other half is mapped by point reflection to the former. The function a is now used for comparing two coins. Let a and a' be two feature functions from two different coins, we define their correlation function by

$$c(\phi) = \int_0^{2\pi} \int_0^1 \delta(a(r,\varphi) - a'(r,\varphi-\phi))\ dr d\varphi,$$

where δ is again some indicator function deciding whether two angles are different or equal. It can be imagined as a delta distribution, for example. So $c(\phi)$ counts how often the angles of the gradients of the two coins coincide at a specific relative angle ϕ. The maximum value of c is defined as our similarity measure

$$k(a,a') = \max_{\phi\in[0,2\pi]} c(\phi) \tag{1}$$

It is well known that the computation of a cross correlation function which is based on the scalar product can be efficiently done in the Fourier domain. But c is not an ordinary scalar product; to get us into the position to apply the Fourier transform we first have to rewrite our correlation function. For any function g there exists the so called convolutional square root $g^{1/2}$, which fulfills the following relation

$$g(x-y) = \int_{\mathbb{R}} g^{1/2}(x-z) g^{1/2}(z-y)\ dz.$$

Using this formula for our indicator function δ we can rewrite the correlation

$$c(\phi) = \int_0^{2\pi} \int_0^1 \int_{\mathbb{R}} \underbrace{\delta^{1/2}(a(r,\varphi)-z)}_{f(r,\phi,z)} \underbrace{\delta^{1/2}(z-a'(r,\varphi-\phi))}_{f'(r,\varphi-\phi,z)}\ dz dr d\varphi.$$

Indeed we introduced an additional integration but now $c(\phi)$ looks like an ordinary correlation of two functions f and f', which can efficiently be computed in the Fourier domain. Let us call $f(r,\phi,z)$ our final feature function and $\tilde{f}(r,k,z) = \int f(r,\varphi,z) e^{-ik\varphi} d\varphi$ its Fourier transform with respect to the angle parameter. The feature with respect the z-parameter can be interpreted as a some kind of indicator function contributing whenever there is a gradient with the specific angle z in the original image.

Using the Fourier features the Fourier representation of $c(\phi)$ looks then

$$\tilde{c}(k) = \int_0^1 \int_{\mathbb{R}} \tilde{f}^*(r,k,z) \tilde{f}'(r,k,z)\ dr dz,$$

where $\cdot^*$ is the complex-conjugate. The last step is just to transform $\tilde{c}(k)$ back into the spatial domain and search for its maxima.

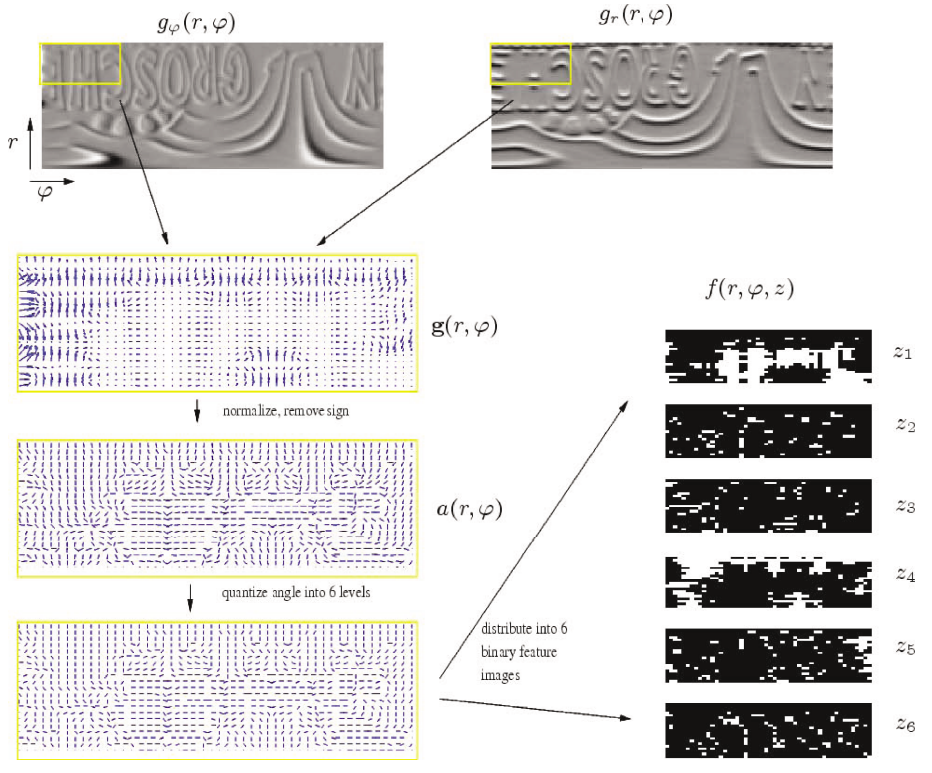


Fig. 3. Sketch of the feature extraction process. After the computation of radial and angular gradients the angle function $a(r,\varphi)$ is computed. For clarity we only visualize the yellow marked upper left corner of the image. In the $a(r,\varphi)$ the arrow-tips are omitted because the sign of the direction does not count anymore. One can see that also regions with relatively low gradient magnitude contain valuable, structured information. In the last step the directions are discretized in six discrete directions. Finally one binary feature image is created for each direction.

3.1 Implementation

Preprocessing and Gradient Computation. Given the segmented image in cartesian coordinates we first applied a blur using the same IIR-filter used for segmentation. To get the polar gradient image we directly sample the gradients from the blurred image with 1024 steps in angular direction and 256 steps in radial direction using bilinear interpolation. We used a stepsize of 1 pixel in the original image for computing the finite differences for the gradient. In radial direction we only sample from 0 to 0.9, i.e. we leave out the outer 10 percent of coin, because it seems that it mostly contains useless information (see also [1]). Then we blur the image again with the IIR-filter and downsample it to the desired size; for the competition we use a size of 256×64. Finally we normalize the gradients and compute the angles according to the equations from above.

Discretizing the Angles and Final Feature Computation. The gradient angle parameter z is discretized in M steps, i.e. we have a stepwidth of $\Delta z = \pi/M$. So the final feature image consists of M binary images of size 256×64. The binary image with number i contains entries whenever $\frac{a(r,\varphi)}{\Delta z}$ falls within the interval $[i, i+1]$ with $0 \leq i < M$. In Figure 3 we visualize the algorithm. Having a closer look at the angle function $a(r, \varphi)$ one can see that also regions with relatively low gradient magnitude contain valuable information.

To become more robust against small gradient changes we additionally use some kind of 'inverse' bilinear interpolation (also called fuzzy histograms, see [4]) to generate the entries. For each entry the two nearest pixel get a contribution depending linearly on the distance to the pixel's center. We also conduct some smoothing in radial direction to be robust against small shifts in radial direction. Such radial shifts usually come from small errors during the segmentation process.

Finally the Fast Fourier Transform (FFT) of the feature function in φ direction can be precomputed to speed up the comparison later, i.e. we compute the FFT of the rows of the binary feature images on the right of Figure 3.

4 Classification

For classification we use a very simple nearest neighbor scheme. This is mainly due to complexity and memory considerations. For example, considering the feature parameters used for the competition, one feature is of size $256 \cdot 64 \cdot 6 \cdot$ sizeof(float) = 392KByte. The training database consists of over two thousand different coin types, so we already need nearly one GigaByte storage. Although, the training set provides a set of different samples per type, we only use one sample for training. But still, a whole scan of those would take about 6 seconds on a *Pentium P4 2.8Ghz* for just one side of the coin, while the benchmark specifications only allowed about 5 seconds to classify one coin, which involves the scan for both sides. To meet this goal we have to restrict to search only in a subset of the two thousand training images. Since we have a good estimate of the radius of the examined coin, the search is only performed in some neighborhood of the estimated radius. We search in a range of $2\Delta = 4mm$ around the estimated radius. Mostly, the search includes around 200 comparison depending on the coin's radius, that is a speed up of a factor of 10 in comparison to an exhaustive search. After the algorithm has determined the nearest neighbors within this range it is checked whether the predicted labels for the front and backside of the coin are consistent. If they do not, the coin is immediately rejected and classified as unknown. Otherwise we compute a prediction confidence C which is based on the similarity scores, thickness, radius and angle pose differences. First we normalize the similarity scores. Imagine that two images are compared that have totally random gradient orientation, then their similarity is expected to be $k = 256 \cdot 64/6 \approx 2730$. Hence, we normalize the similarity score by $k' = k - 2730$. We further compute the thickness differences Δt_1 and Δt_2 for the front and backside of the coin to its matched partners from the training database. And we

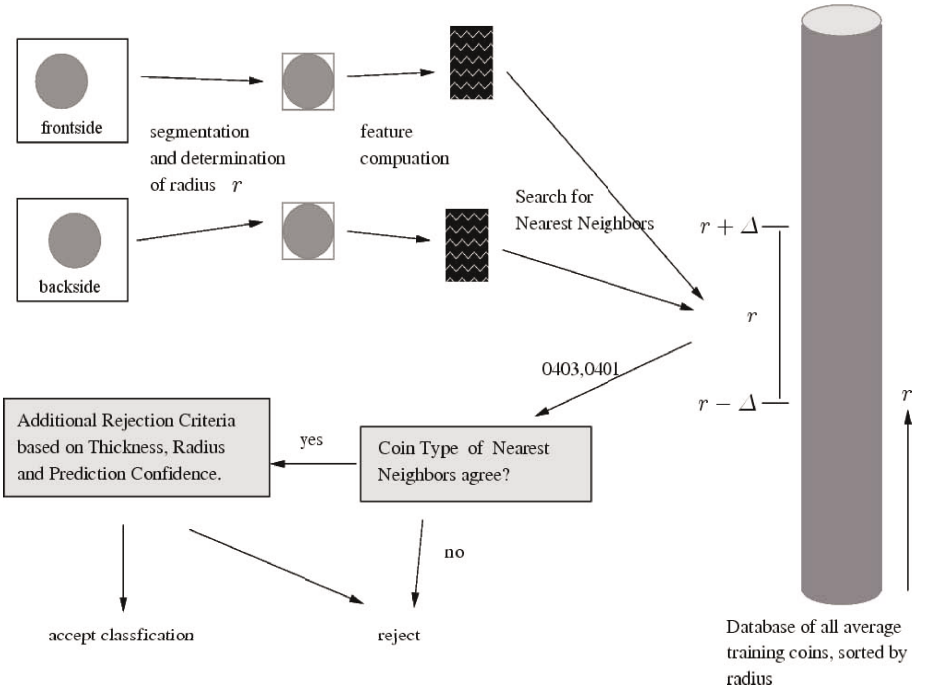


Fig. 4. Complete workflow of the classification system. After segmentation of the coins the features $\tilde{f}(k, r, z)$ according to section 3 are computed. Then, nearest neighbors are searched with respect to the similarity measure (1). The search is only performed within a neighborhood of the estimated radius, not on the whole database. If the nearest neighbors of both coin sides agree the coin is a possible candidate for acceptance, otherwise it is immediately rejected. Finally additional rejection criteria are checked and some classes get a special treatment.

do the same for the radius. Due to the construction of the coin acquisition system front and backside of the coin should have a fixed rotational pose relation. We found that the relation is slightly unreliable, but we also included it with a small weight in our confidence score,

$$C = k_1' k_2' \exp\left(-\frac{(\Delta r_1 + \Delta r_2)}{0.17\ mm} - \frac{(\Delta t_1 + \Delta t_2)}{0.07\ mm} - \frac{(\Delta \phi)}{10^\circ} \right).$$

If this confidence is below 10^{-5} the coin is rejected. Additionally, we will make a reject if the thickness difference is above $0.25mm$ or the $\Delta\phi$ is above 70°. Besides, all this additional reject criteria do not have too much influence on the overall performance, the most powerful rejection criteria is definitely the consistency of the votes for the front and backside of the coin. In Figure 4 we give a rough overview over the complete classification system.

5 Experiments

We conducted some experiments to get good settings for the parameters. We found that for the angular/radial resolution 256×64 is a good trade-off betwees accuracy and speed of the classification. In general one can say that the higher the resolution the better the results, while the ratio $256/64 = 4$ seems to be optimal and should be kept fix. As already mentioned we use 6 discritization steps for the z-resolution.

Table 1. Experimental results for all three tranches (left) and the competetion results (right). RC is the number of classes which got at least one correct classification. (T/F)(P/N) means True/False Positives/Negatives. TP+TN gives the total number of correct classification. FN+TN is the total number of rejects, i.e. the number of coins classified as unknown. FP gives the number of unknown coins, which are classified to some known class and the number of known coins classified to a wrong known class. The assessment score is calculated by score = RC*25 + TP+TN - 100*(FP).

Tranche	1A (no rej.)	1A	1B	1C	**FR**	MA
RC	322	320	309	398	**339**	278
TP+TN (%)	97.64	97.27	97.20	94.31	**97.24**	67.31
FN+TN (%)	5.20	5.64	5.94	9.42	**5.18**	32.50
FP (%)	0.06	0.02	0.02	0.03	**0.0**	2.21
Ass. Score	17211	17527	17245	19081	**18199**	-8419

In Table 1 we show the obtained results on the CIS benchmark dataset. The system shows very good performance, classification rate are mostly above 97% and the false positive rate is very low. In the first column results for tranche 1A with no additional reject criteria are shown. Obviously the additional criteria improve the system; less false positives and only marginal shrinkage of the classication rate can be observed, and hence also a higher assessment score is obtained. Further we show results for tranches 1B and 1C with the additional reject criteria. On all tranches, together 30000 coins, we have in total 7 coins which are wrongly classified to be known, while they are labeled as unknown. Six of these coins are very similar to known coin classes or are wrongly labeled. The results for the final competition results are also reported (details in [9]). Our results (FR) are compared to an approach based on invariant features (MA) [8].

6 Conclusions

We presented a coin recognition system which is based on gradient directions only. The results show that the directional information is enough to build a reliable classification system while the system is very robust to illumination and contrast changes. We have shown that the demand of a very low false positive rate is possible to reach. There might be several improvements of the system. Further

fitting of the parameters like resolution, oversampling multiplier or smoothing width may improve the accuracy of the system. More sophisticated reject criteria and confidence values could also help to avoid false positives. Unfortunately the test set sizes of ten thousand coins are still to small to validate false positive rates of 0.01 percent.

References

1. Nölle, M., Penz, H., Rubik, M., Mayer, K., Holländer, I., Granec, R.: Dagobert - A new Coin Recognition and Sorting System. In: Proceedings of the 7th Internation Conference on Digital Image Computing - Techniques and Applications (DICTA'03), Sydney, Australia (2003)
2. Ballard, D.H.: Generalizing the hough transform to detect arbitrary shapes. Pattern Recognition, 13–2 (1981)
3. Schulz, J., Schmidt, T., Ronneberger, O., Burkhardt, H.: Fast Scalar and Vectorial Grayscale based Invariant Features for 3D Cell Nuclei Localization and Classification. In: Franke, K., Müller, K.-R., Nickolay, B., Schäfer, R. (eds.) Pattern Recognition. LNCS, vol. 4174, Springer, Heidelberg (2006)
4. Siggelkow, S., Burkhardt, H.: Improvement of Histogram-Based Image Retrieval and Classification. In: Proceedings of the International Conference on Pattern Recognition, vol. 3, pp. 367–370 (2002)
5. Haber, Ramoser, Mayer, Penz, Rubik: Classification of coins using an eigenspace approach. Pattern Recognition Letters 26(1), 61–75 (2005)
6. Fukumi, M., Omatu, S., Takeda, F., Kosaka, T.: Rotation-invariant neural pattern recognition system with application to coin recognition. IEEE Transactions in Neural Networks 3(2), 272–279 (1992)
7. Adameck, M., Hossfeld, M., Eich, M.: Three color selective stereo gradient method for fast topographic recognition of metallic surfaces. In: Proceedings of Electronic Imaging, Science and Technology, Machine Vision Application in Industrial Inspection XI, vol. SPIE 5011, pp. 128–139 (2003)
8. van der Maaten, L., Boon, P.: COIN-O-MATIC: A fast system for reliable coin classification. In: MUSCLE CIS Coin Recognition Competition Workshop (2006), `http://muscle.prip.tuwien.ac.at`
9. Nölle, M., Rubik, M., Hanbury, A.: Results of the MUSCLE CIS Coin Competition 2006. In: MUSCLE CIS Coin Recognition Competition Workshop (2006), `http://muscle.prip.tuwien.ac.at`