SCHEDAE INFORMATICAE
VOLUME 19 2010

# Investigation of Normalization Techniques and Their Impact on a Recognition Rate in Handwritten Numeral Recognition

WIESŁAW CHMIELNICKI[1], KATARZYNA STĄPOR[2]
[1]Faculty of Physics, Astronomy and Applied Computer Science,
Jagiellonian University, Reymonta 4, 30-059 Kraków,
e-mail: *wieslaw.chmielnicki@uj.edu.pl*
[2] Institute of Computer Science, Silesian Technical University,
Akademicka 16, 44-100 Gliwice

**Abstract.** This paper presents several normalization techniques used in handwritten numeral recognition and their impact on recognition rates. Experiments with five different feature vectors based on geometric invariants, Zernike moments and gradient features are conducted. The recognition rates obtained using combination of these methods with gradient features and the SVM-rbf classifier are comparable to the best state-of-art techniques.

**Keywords:** handwritten numeral recognition, normalization techniques, SVM classifier, feature vectors, OCR, geometric invariants, Zernike moments, gradient features.

## 1. Introduction

The recognition of handwritten numerals by computer has been a subject of intensive research for about last fifty years. The problem, which is very simple for almost every human, is extremely complicated for the machine. Hundreds of scientist developed many sophisticated systems but computers are still unable to compete with human capabilities. The main reason that causes recognition of characters and numerals such a complex problem is the variety of writing styles. Each character or numeral can look differently depending on a writing person, light or heavy prints, varying levels of care, etc.

A typical OCR system usually consists of three main processing stages: preprocessing, feature extraction/selection and recognition using a classifier. Preprocessing is a very important factor for the next two recognition stages: feature extraction and classification. It consists of a sequence of operations applied to the images to prepare them for feature extraction. Usually we start with noise removal and/or smoothing [40], document skew correction [41], normalization [2], slant correction [42]. Depending on a feature extraction method additional preprocessing steps might be required such as thinning [43] or contour analysis.

Feature extraction consists of large variety of techniques which allow us to represent an image as a vector of values (features). These features can be based for example on geometric moment invariants [1], Zernike moments [7] or gradients [9, 10]. At this stage a feature selection algorithm can be applied to reduce the size of the input feature vector to avoid the so-called curse of dimensionality problem.

There is also a large number of classifiers. Beginning from parametric statistical classifiers, neural networks, SVMs (Support Vector Machines) and finishing on hybrid classifiers. At each stage we can choose parameters which could affect the final classification performance.

This paper focuses on recognition of unconstrained handwritten numerals and especially on the normalization techniques. It shows the impact of image normalization on the classification performance. There are several normalization methods (described in [2]) which have been implemented to use in this research. The experiments with these methods show how different normalization algorithms influence the final classification performance. Five different feature vectors of three types (geometric invariants, Zernike moments and gradient features) are used in the experiments. The influence of normalization on each of these vectors is shown. In this research the SVM classifier based on the statistical learning theory of Vapnik [25] was used.

The MNIST (modified NIST) digit database was used in the experiments. The NIST database was collected from specially designed forms filled by US Census Bureau employees and High school students. The number of training

samples and test samples are 60 000 and 10 000, respectively. This database is widely used in various character recognition researches.

In recent years many feature extraction methods for handwritten numeral recognition have been proposed. A survey of these methods can be found in [50]. Additionally, some new features such as: stroke features [20], curvature features [10], local structure features [21] or structural and concavity features [44] are used to enhance a recognition ratio.

There are also many approaches to the classification task of handwritten characters, like: statistical techniques [22], neural networks and Support Vector Machines (SVMs) [23]. The parametric or non-parametric statistical classifiers, the linear discriminant function (LDF), the quadratic discriminant function (QDF), the nearest-neighbour (1-NN) and k-NN classifiers, the Parzen window classifier, etc are used. They can be modified: for example a modified quadratic discriminant function was proposed by Kimura et al. [24, 26]. Neural networks include the multilayer perceptron (MLP), the radial basis function (RBF) network and the polynomial classifier. Several of these classifiers have been evaluated in [27].

There are three main databases: CENPARMI [16], CEDAR [28], and MNIST [29] used in handwritten characters recognition. They have been widely used for validating the recognition performance. The CENPARMI digit database was released by the Concordia University. It contains 6000 digits divided into a train set (4000 images – 400 samples per class) and a test set (2000 images – 200 samples per set). The CEDAR digit database was released by CEDAR, SUNY Buffalo. The training data set contains 18 468 digit images. The test data set contains 2711 digit images. The last database – MNIST (modified NIST) was extracted from the NIST special databases SD3 and SD7. In this database the sizes of training and test sets are 60 000 and 10 000 images, respectively.

**Tab. 1.** Results on CENPARMI database

| Method | Recognition rate |
|---|---|
| Franke [33] | 97.60% |
| Suen et al. [30] | 98.85% |
| Liu et al. [31] | 98.45% |
| Gader et al. [34] | 98.30% |
| Liu et al. [2] | 99.15% |

Below the results achieved on these databases are presented. In the following tables the best recognition rates obtained using corresponding databases are shown. The recognition rate is defined as the number of correctly recognized images/the total number of images of the test dataset.

The results achieved using the CENPARMI database are presented in Tab. 1. The recognition rates vary from 97.6% to 99.15%. Some of these results have been received using multiple classifiers for example Suen et al. [30], but even better results 99.15% can be achieved using a single classifier as in [2].

**Tab. 2.** Results on CEDAR database

| Method | Recognition rate |
|---|---|
| Suen et al. [30] | 99.77% |
| Liu et al. [28] | 98.87% |
| Cai et al. [35] | 98.4% |
| Oh et al. [36] | 98.73% |

The list of results obtained using the CEDAR database is presented in Tab. 2. The recognition rates are in the range from 98.4% to 99.77%. The best result was achieved by combining multiple classifiers in [30], the best result using a single classifier was 98.87% [28].

The third database, MNIST, is most widely used for evaluation of recognition algorithms. Some results on this database are presented in Tab. 3. Here the recognition rates vary from 98.3% to 99.41%. The highest accuracy was given by Teow and Loe using SVC on direction and stroke-end features [8]. The same result 99.41% was achieved by Liu et al. [2] on the NIST database (MNIST is a modified NIST).

**Tab. 3.** Results on MNIST database

| Method | Recognition rate |
|---|---|
| Teow et al. [8] | 99.41% |
| Mayraz et al. [37] | 98.3% |
| Dong et al. [38] | 99.01% |
| Belongie et al. [39] | 99.37% |
| Liu et al. [2]* | 99.41% |

The rest of this paper is organized as follows. Section 2 introduces the normalization strategies, Section 3 describes features extraction methods used in this paper. Section 4 shortly describes foundations of the SVM classifier. Section 5 presents experimental results, while Section 6 conclusions and future work.

## 2. Normalization techniques

Normalization is a process that changes different image parameters to obtain more convenient values. Normalization techniques are used generally to reduce the within-class variation. For example the intensity normalization tries to equalize intensity of the character images, the perspective transformation may correct the imbalance of the character width [32], the moment normalization tries to correct the rotation or slant [64] and the ratio normalization changes the character aspect ratio. Normalization is considered to be the most important preprocessing factor for character recognition [55].

In most experiments considering feature extraction and classification of characters a square standard plane with the fixed $N \times N$ dimensions is used. All original character images, usually of different sizes, are mapped onto this plane. From this standard plane a feature vector is extracted. The algorithms described below are used to perform these mappings.

Let $W_1$ and $H_1$ denote the width and the height of the original character image, respectively. Then the aspect ratio of the original image is defined as:

$$R_1 = \frac{\min(W_1, H_1)}{\max(W_1, H_1)}. \tag{1}$$

Similarly, let $W_2$ and $H_2$ denote the width and the height of the normalized image, respectively. Then the aspect ratio the normalized image is defined as:

$$R_2 = \frac{\min(W_2, H_2)}{\max(W_2, H_2)}. \tag{2}$$

Both $R_1$ and $R_2$ are in the range [0, 1), as it follows from Eqs (1) and (2).

In the described experiments an aspect ratio adaptive normalization (ARAN) strategy [2] is used. In this strategy the normalized aspect ratio $R_2$ is calculated based on the original aspect ratio $R_1$ using different mapping functions. So using ARAN normalization a character image is fitted into a new normalized plane $H_2 \times W_2$, and then this plane is shifted to overlap the standard plane. Dimensions of the normalized plane are calculated as follows. It is assumed that one dimension of the normalized plane fills one dimension of the standard plane: $N = max(H_2, W_2)$, the other dimension is calculated using the aspect ratio $R_2$ and then centred on the standard plane (see Fig. 1).

The transformation to the standard plane can be described using a coordinate mapping. Let us denote the original image as $f(x, y)$ and the normalized image as $g(x', y')$. Then the normalized image can be generated by coordinate mapping $g(x', y') = f(x, y)$. We can use the forward mapping or the backward mapping. These mappings are given by:

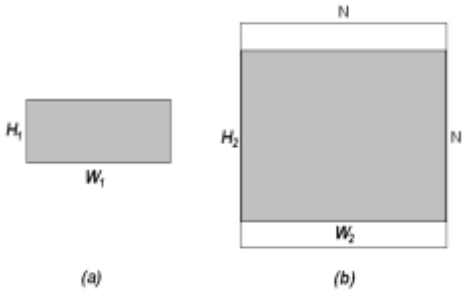


**Fig. 1.** (a) An original image, (b) A normalized image on the standard plane

$$x' = x'(x, y), \quad y' = y'(x, y)$$

and

$$x = x(x', y'), \quad y = y(x', y')$$

respectively, where *x'*, *y'* are the coordinates of the normalized image and *x*, *y* are the coordinates of the original image. The mappings used in the experiments are presented in Tab. 4.

When the forward mapping is used, *x* and *y* are discrete, but *x'* and *y'* can be real values. Similarly, when the backward mapping is used, then *x'* and *y'* are discrete, but *x* and *y* can be real values. Moreover, in the forward mapping, the mapped coordinates *x'*, *y'* usually do not fill all pixels in the normalized plane. So the coordinate discretization or pixel interpolation is necessary.

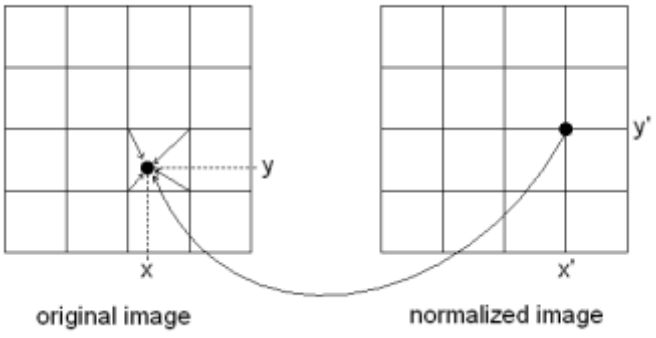


**Fig. 2.** An example of the backward mapping

The discretization algorithm is quite simple. The mapped coordinates ($x'$, $y'$) or ($x$, $y$) are approximated by the closest integer numbers ([$x'$], [$y'$]) or ([$x$], [$y$]). Then in case of the forward mapping, the discrete coordinates ($x'$, $y'$) scan the pixels of the original image and the pixel value $f(x, y)$ is assigned to all pixels ranged from ([$x'(x, y)$], [$y'(x, y)$]) to ([$x'(x+1, y+1)$], [$y'(x+1, y+1)$]). In case of the backward mapping discretization is trivial.

In the described experiments grey-scaled images are used, so the interpolation algorithm must be used. In case of the backward mapping the mapped pixel ($x$, $y$) is surrounded by four discrete pixels. The grey level $g(x', y')$ is a weighted combination of the four pixel values. It is graphically presented in Fig. 2. In the forward mapping every pixel in the original image and the normalized image are treated as squares of a unit area. The unit square of the original image is mapped to a rectangle in the normalized plane. It is illustrated in Fig. 3. Next, each unit square overlapping the rectangle is given a grey level value proportional to the overlapping area.

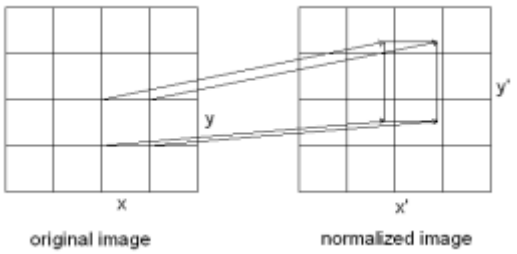


**Fig. 3.** An example of the forward mapping

Two mappings used in the experiments are presented in Tab. 4. When using the moment-based normalization method one dimension may go beyond the standard plane. In this case the image part outside the standard plane is cut off.

**Tab. 4.** Normalization methods used in experiments

| | Forward mapping | Backward mapping |
|---|---|---|
| Linear mapping | $x' = \alpha x$<br>$y' = \beta y$ | $x = x'/\alpha$<br>$y = y'/\beta$ |
| Moment mapping | $x' = \alpha(x - x_c) + x'_c$<br>$y' = \beta(y - y_c) + y'_c$ | $x = (x' - x'_c)/\beta + x_c$<br>$y = (y' - y'_c)/\beta + y_c$ |

where:

$$\alpha = W_2 / W_1, \ \ \beta = H_2 / H_1,$$

the centre of gravity of the image is given by

$$x_c = m_{10} / m_{00}, \ \ y_c = m_{01} / m_{00},$$

$m_{pq}$ denotes the geometric moments:

$$m_{pq} = \sum_x \sum_y x^p y^q f(x, y),$$

$x'_c$, $y'_c$ denote the geometric centre of the normalized plane given by:

$$x'_c = W_2 / 2, \ y'_c = H_2 / 2.$$

In the described experiments, all normalization functions are implemented by the backward mapping. Fig. 4 shows samples of the normalized images, corresponding to all normalization functions. The size of the standard plane is 32 × 32.

In the described experiments there are several normalization methods implemented as described above. The used normalization functions are listed in Tab. 5.

**Tab. 5.** List of normalization functions used in the experiments

| Symbol | Description | Aspect ratio |
|---|---|---|
| N0 | Linear normalization with fixed aspect ratio | $R_2 = 1$ |
| N1 | Linear normalization with preserved aspect ratio | $R_2 = R_1$ |
| N2 | Linear normalization with square root ratio | $R_2 = \sqrt{R_1}$ |
| N3 | Linear normalization with cube root ratio | $R_2 = \sqrt[3]{R_1}$ |
| N4 | Linear normalization with fixed aspect ratio* | $R_2 = 0.9$ |
| N5 | Linear normalization with square root of sine of aspect ratio | $R_2 = \sin(\pi / 2R_1)$ |
| N6 | Moment normalization with preserved aspect ratio | $R_2 = R_1$ |
| N7 | Moment normalization with square root ratio | $R_2 = \sqrt{R_1}$ |
| N8 | Moment normalization with cube root ratio | $R_2 = \sqrt[3]{R_1}$ |
| N9 | Moment normalization with fixed aspect ratio* | $R_2 = 0.9$ |
| N10 | Moment normalization with square root of sine of aspect ratio | $R_2 = \sin(\pi / 2R_1)$ |

* The aspect ratio obtained using test procedure from range [0.4, 1).

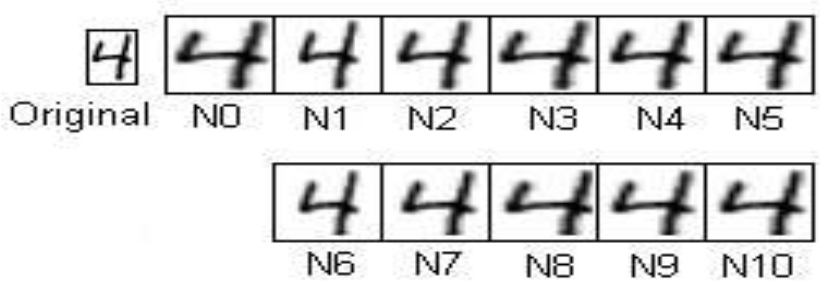


**Fig. 4.** Normalized images

## 3. Feature extraction

In the described experiments three types of features are used: geometric moment invariants, Zernike moments and gradient features. The moment invariants are known to be invariant under rotation, translation, scaling and reflection. The Zernike moments are noise resilient. The gradient features are easy to extract and give the high performance and discriminative power as well.

### 3.1. Geometric moment invariants

First type of features which were used are geometric moment invariants. These features extract global properties of the image such as the shape area, the centre of the mass, the moment of inertia, and so on. In these experiments a feature vector similar to presented in [46] is used, but modified and extended to a 98D vector. Given a grey-scale image of the size $M \times N$, the regular moments of order $(p + q)$ are defined as:

$$m_{pq} = \sum_{i=1}^{N} \sum_{j=1}^{M} x_i^p y_j^q f(x_i, y_j).$$

From the above translation-invariant central moments can be obtained by placing the origin in the centre of gravity.

$$\mu_{pq} = \sum_{i=1}^{N} \sum_{j=1}^{M} (x_i - x)^p (y_j - y)^q f(x_i, y_j),$$

where

$$x = \frac{m_{10}}{m_{00}}, \quad y = \frac{m_{01}}{m_{00}}.$$

Hu showed that:

$$\eta_{pq} = \mu_{pq} / \mu_{00}^{(\frac{p+q}{2}+1)}, \quad p+q \geq 2$$

are scale-invariant.

Finally, rotation-invariant feature can be constructed. In this paper seven invariants were used as follows:

$$\Phi_1 = \eta_{20} + \eta_{02},$$
$$\Phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2,$$
$$\Phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2,$$
$$\Phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2,$$
$$\Phi_5 = (\eta_{30} - \eta_{12})(\eta_{30} + \eta_{12})((\eta_{30} + \eta_{21})^2 - 3(\eta_{21} + \eta_{03})^2) +$$
$$(3\eta_{21} - \eta_{30})(\eta_{21} + \eta_{03})((\eta_{30} + \eta_{21})^2 - (\eta_{21} + \eta_{03})^2),$$
$$\Phi_6 = (\eta_{20} - \eta_{02})((\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2) + 4\eta_{11}(\eta_{30} + \eta_{21})(\eta_{21} + \eta_{03}),$$
$$\Phi_7 = (3\eta_{21} - \eta_{03})(\eta_{03} + \eta_{12})((\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2) +$$
$$(3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})(3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2).$$

As these features are rotation-invariant there will be a problem with recognizing some numerals, i.e. 6 and 9 or 2 and 5. As a matter of fact the recognition rate using these seven features was only about 45%. To avoid this problem the image was divided into 4 and 9 square regions and all seven geometric invariants were extracted from every region. This gives a (4 + 9) * 7 + 7 = 98D feature vector. This feature vector is denoted as GMI.

### 3.2. Zernike moments

A Zernike moments concept was first introduced by Teague in 1980 [49]. Compared to the geometry moment invariants Zernike moments are computationally expensive, but have several advantages: they are orthogonal, rotation invariant and noise resilient. Additionally, they have one interesting feature, the original image can be reconstructed from these moments. They have been used to binary pictures because they are not invariant due to

contrast. This drawback can be easily avoided using the grey-scale normalization. Complex Zernike moments are constructed using a set of complex polynomials which form a complete orthogonal basis set defined on the unit disc. These polynomials are defined as below:

$$V_{nm}(x,y) = R_{nm}(x,y)e^{jm\tan^{-1}(y/x)},$$

where

$$j = \sqrt{-1},\ \ n \geq 0,\ \ n - |m| \text{ is even}$$

and

$$R_{nm}(x,y) = \sum_{s=0}^{(n-m)/2} \frac{(-1)^s (x^2+y^2)^{n/2-s}(n-s)!}{s!\left(\frac{n+|m|}{2}-s\right)!\left(\frac{n-|m|}{2}-s\right)!}.$$

Then Zernike moment of order $n$ and repetition $m$ is defined as:

$$A_{nm} = \frac{n+1}{m}\sum\sum f(x,y)(V_{nm}(x,y))^*, \quad x^2+y^2 \leq 1, \tag{3}$$

where * denotes a complex conjugate operator, $n - |m|$ is even and $|m| \leq n$.

It is interesting that the original image can be reconstructed using the formula:

$$f(x,y) = \lim_{N\to\infty} \sum_{n=0}^{N} \sum_{m} A_{nm} V_{nm}(x,y), \tag{4}$$

where the sum is taken for all $|m| \leq n$ and $n - |m|$ is even.

The amplitudes of Zernike moments $|A_{nm}|$ are rotation invariant. Invariance to the scale and translation can be obtained by shifting and scaling the image before the computation of Zernike moments. The normalization algorithms used in these experiments guarantee that all images are shifted and scaled.

There are two feature vectors based on Zernike moments used. The first vector ZM1 consists of all 47 first amplitudes of Zernike moments from $Z_{0,0}$ to $Z_{12,12}$, and the second vector ZM2 consists of 24 amplitudes chosen only as described in [7], i.e: $Z_{0,0}$, $Z_{2,0}$, $Z_{3,1}$, $Z_{3,3}$, $Z_{4,0}$, $Z_{4,2}$, $Z_{5,1}$, $Z_{5,3}$, $Z_{5,5}$, $Z_{6,0}$, $Z_{7,1}$, $Z_{7,3}$, $Z_{7,5}$, $Z_{8,4}$, $Z_{8,6}$, $Z_{9,5}$, $Z_{9,7}$, $Z_{10,2}$, $Z_{10,4}$, $Z_{11,1}$, $Z_{11,5}$, $Z_{11,7}$, $Z_{12,0}$.

#### 3.2.1. Fast algorithm to compute Zernike moments

The algorithm computing Zernike moments with the use of Eq. (3) will be very inefficient. As a matter of fact it is useless according to the database size. So there must be found a far more efficient method. Let us notice that under polar coordinates the above formula can be expressed as:

$$A_{nm} = \frac{n+1}{\pi} \sum_x \sum_y R_{nm}(r) e^{-jm\theta} f(r,\theta), \tag{5}$$

where

$$r = \sqrt{x^2 + y^2}, \quad \theta = \tan^{-1}(x/y)$$

and

$$R_{nm}(r) = \sum_{s=0}^{(n-m)/2} \frac{(-1)^s (n-s)!}{s!(\frac{n+|m|}{2}-s)!(\frac{n-|m|}{2}-s)!} r^{n-2s}. \tag{6}$$

Then imaginary and real components of Zernike moment can be calculated as:

$$C_{nm} = \frac{2n+2}{\pi} \sum_x \sum_y R_{nm}(r) \cos(m\theta) f(r,\theta) \tag{7}$$

and

$$S_{nm} = \frac{2n+2}{\pi} \sum_x \sum_y R_{nm}(r) \sin(m\theta) f(r,\theta). \tag{8}$$

The character images are discrete and usually relatively small (in these experiments 32 x 32) so the number of different values of $r$ is relatively small. Let us denote the pixel in the centre of the image as level 0, and the next 8 neighbouring pixels as level 1 and so on. On each level there will be *level*+1 different values of $r$. The number of levels will be:

$$1 + 2 + \ldots + \lfloor n/2 \rfloor + (\lfloor n/2 \rfloor + 1).$$

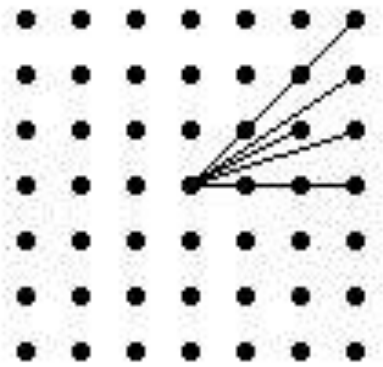

**Fig. 5.** All possible values of $r$ for a 7 x 7 image

The total number of different values of $r$ will be: $1/8(n^2 + 6n + 8)$. Usually only a few dozen of first Zernike moments are used, so all values of $R_{nm}(r)$ can be computed using Eq. (5) for all possible values of $r$, $m$, $n$. For example in the described experiments there are 32 x 32 images and 47 first Zernike moments are used. It gives 153*47 = 7191 different values for all images.

Precomputing all these values has a great impact on the algorithm efficiency because all values of $R_{nm}$ are calculated only once for all images (the MNIST database used in the experiments has 60 000 + 10 000 = 70 000 images). Values of $sin(m\theta)$, $cos(m\theta)$ (or even $R_{nm}(r)sin(m\theta)$, $R_{nm}(r)cos(m\theta)$) can be precomputed as well.

### 3.3. Gradient features

The gradient features can be easily used to grey-scale images and are robust against image noise and edge direction fluctuations. Additionally, the gradient can be computed by using the Sobel operator, which has two masks for the gradient components in horizontal and vertical directions. So it can be efficiently extracted from the image. The gradient gives us the magnitude and the direction of the greatest change in intensity in the neighbourhood of a pixel. The Roberts [10] and Kirsh [65] operator have also been used in the literature.

| | | |
|---|---|---|
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

| | | |
|---|---|---|
| 1 | 2 | 1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

**Fig. 6.** Sobel masks used to compute gradients

The Sobel operator is used to compute gradient components as follows:

$$g_x(x,y) = f(x+1,y-1) + 2f(x+1,y) + f(x+1,y+1) - f(x-1,y-1) - 2f(x-1,y) - f(x-1,y-1) \tag{9}$$

$$g_y(x,y) = f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1) - f(x-1,y-1) - 2f(x,y-1) - f(x+1,y-1). \tag{10}$$

Then, the gradient magnitude is calculated as:

$$A(x,y) = \sqrt{g_x^2(x,y) + g_y^2(x,y)} \tag{11}$$

and the gradient direction as:

$$\theta(x,y) = \tan^{-1}\left(\frac{g_y(x,y)}{g_x(x,y)}\right).$$

The complete gradient map may contain some noise information. Especially when grey-scale images are used. To avoid these spurious gradients a simple filtering algorithm is proposed – an adaptive gradient thresholding. In the first step the average gradient magnitude is computed over the whole image and then this value is used to filter our gradient map. Formally:

$$g_{avr} = \frac{\sum_x \sum_y A(x,y)}{M \times N},$$

where *M, N* are dimensions of the image. Then

$$\bar{A}(x,y) = \begin{cases} A(x,y) & A(x,y) \geq A_{avr} \\ -1 & A(x,y) < A_{avr} \end{cases}$$

and

$$\bar{\theta}(x,y) = \begin{cases} \theta(x,y) & A(x,y) \geq A_{avr} \\ -1 & A(x,y) < A_{avr} \end{cases}.$$

The gradient directions are real values from range [0, 360). To extract a feature vector they are quantized into a small number of integer values. There are 12 integer values used representing gradient scopes: [0, 30), [30, 60), [60, 90) and so on. Next, the gradient map is divided into 4 x 4 parts. Then, a percentage of pixels with the direction of gradient quantized to value $K$ = 1, 2, ..., 12 is computed in each part. Hence the total number of features will be $4 \times 4 \times 12 = 192$D. This feature vector is denoted as GF.

The second feature vector denoted as GFC is also based on gradient features. There are 10 crossing line features added to the previous vector GF. The crossing line features are extracted in the following steps. First, the centre of gravity of the image is found, then the horizontal and vertical line are drawn through this point, and finally two extra lines on each side of the horizontal and vertical line are added with equal margins. The crossing line

feature is the number of intersection points with the image. For example for the numeral shown in Fig. 7 there are two vectors: (1,1,1,1,1) and (1,2,2,2,1).

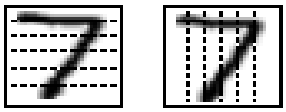


**Fig. 7.** Crossing line features (a) horizontal, (b) vertical

## 4. The SVM classifier

The Support Vector Machine (SVM) has been proposed by Vapnik in [25]. The SVM technique has been used in different application domains and has outperformed the traditional techniques in terms of generalization capability. Contrary to the traditional techniques which try to minimise the empirical risk (the classification error on the training data) SVM minimises the structural risk (the classification error on data never seen before).

The classification task is to predict whether a test sample belongs to one of two classes. In a feature space this corresponds to finding a hyperplane which separates these two classes. There is an infinite number of such hyperplanes, so among the possible choices, the SVM classifier selects the one for which the distance of the hyperplane from the closest feature vectors (the "margin") is as large as possible. This hyperplane is called an optimal separating hyperplane. Let us consider a classifier whose decision function is given by:

$$f(x) = sign(x^T w + b),$$

where $x$ denotes a feature vector and $w$ is a weight vector. The problem is separable when there exist $w$ and threshold $b$ such that:

$$y_i(x_i^T w + b) \geq 1, \quad i = 1,2,\ldots,m.$$

To maximize the margin we must minimize

$$1/2\|w^2\|.$$

This problem leads to a so-called dual optimization problem, which is

$$L_D = \sum_{i=1}^{N} \alpha_i - 1/2 \sum_{i,j}^{N} \alpha_i \alpha_j (x_i^T x_j),$$

such that

$$\alpha_i > 0,\ i = 1,2,\ldots,N \text{ and } \sum_{i=1}^{N} \alpha_i y_i = 0.$$

This leads to a hyperplane decision function:

$$f(x) = sign(\sum_{\text{support vectors}} y_i\ \alpha_i (x_i^T x) + b), \tag{12}$$

where $x_i$ are support vectors with Lagrangian non-zero multipliers $\alpha_i$.

The support vectors are the feature vectors which lie on the margins. This is an advantage of this approach because only a small number of vectors is used to compute a resulting classifier.

In a real life problem it is unlikely that a hyperplane will exactly separate the data. To deal with this problem the soft margin hyperplanes are used. A set of variables $\xi_i$ representing errors (i.e. the vectors which lie inside the margin) and a parameter $C$ which determines a trade-off between margin maximization and error minimization are introduced.

In Eq. (12) a dot product of the input vectors is used. So we can apply some trick to calculate the dot product of the vectors in the feature space using a kernel function. It allows us to create a decision function that is non-linear in the input space, but is linear in the feature space, i.e.:

$$f(x) = sign(\sum_{\text{support vectors}} y_i\ \alpha_i K(x_i, x) + b), \tag{13}$$

where $K(x_i, x)$ is a kernel function. Typical kernel functions are:

1. Linear kernel: $K(x_i, x) = x^T x_i$,
2. Polynomial kernel: $K(x_i, x) = (\gamma x^T x + c)^d$,
3. RBF (Radial basis kernel) Kernel: $K(x_i, x) = -\gamma \|x - x_i\|^2, \quad \gamma > 0$,
4. Gausian RBF Kernel: $K(x_i, x) = \exp(-\|x - x_i\|)^2 / 2\sigma^2$,
5. Sigmoid kernel: $K(x_i, x) = \tanh(\gamma x^T x_i + c)$.

The last problem which must be solved is that handwritten numeral recognition is a multi-class problem and the SVM is a binary classifier. There are two commonly used solutions. The first is WTA (winner takes all) strategy. In this approach we build N classifiers for N classification problems: one class versus all other classes. Another approach is to build $n(n–1)/2$ classifiers for each pair of classes, then use MVS (majority voting scheme) strategy.

## 5. Experimental results

In this section recognition accuracies obtained in the experiments are presented. The results are presented in Tab. 6. In the rows there are five different feature vectors (described in Sect. 3) and in the columns N0 – N10 are normalization methods used (see Tab. 2). In the first column WN there is accuracy obtained without any normalization.

The handwritten digit database MNIST described in Section 1 is used in these experiments. Some images of the training dataset are shown in Fig. 8. This database is divided into two datasets: the training dataset including 60 000 samples and the test dataset including 10 000 samples. As a SVM-rbf classifier is used in the experiments there is an extra dataset necessary for validation purposes (to find $C$ and $\gamma$ parameters). So the training dataset was divided into two sets: 50 000 samples for the training set and 10 000 samples for the validation set.

For these three datasets feature vectors are generated using normalization methods listed in Tab. 5. For normalization methods N4 and N9 a special procedure was used to find the best ratio $R_2$. The 12 feature vectors are generated using aspect ratios from 0.4 to 0.95 with step 0.05. The best result (for aspect ratio 0.9) is presented in Tab. 6. The extra feature vector set is generated for original images.

**Fig. 8.** Examples from MNIST database

Before using the classifier all feature vectors are linearly scaled into [–1, 1] range. The main advantage of using this scaling is to avoid attributes in greater numeric ranges dominating those in smaller numeric ranges. Another advantage is to avoid numerical difficulties during the calculation. Because kernel values usually depend on the inner products of feature vectors, e.g. the linear kernel and the polynomial kernel, large attribute values might cause numerical problems.

In Tab. 6 the results of the experiments are shown. There are feature vectors in rows and normalization methods in columns.

**Tab. 6.** Recognition rates obtained on different feature vectors

| | WN | N0 | N1 | N2 | N3 | N4 | N5 | N6 | N7 | N8 | N9 | N10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| GMI | 79.31 | 82.58 | 83.49 | 85.42 | 85.18 | 85.52 | 84.72 | 84.93 | 85.44 | **85.47** | 85.03 | 84.99 |
| ZM1 | 84.10 | 91.42 | 92.84 | 91.12 | 93.54 | 93.94 | 91.80 | 92.90 | **94.33** | 94.22 | 94.19 | 94.05 |
| ZM2 | 86.24 | 92.20 | 93.91 | 94.02 | 94.21 | 93.29 | 92.11 | 93.45 | **94.79** | 94.62 | 94.77 | 94.31 |
| GF | 94.52 | 96.78 | 96.83 | 96.98 | 98.01 | 97.61 | 97.83 | 97.70 | 98.76 | 98.61 | **98.77** | 98.72 |
| GFC | 95.24 | 97,86 | 97.65 | 98.21 | 98.48 | 98.32 | 98.47 | 98.12 | **99.16** | 98.82 | 98.98 | 99.06 |

Geometric moments invariants are the poor feature vector. But as can be seen in [46] and in our experiments the results are close to the best features vectors based on gradient and directional features. Possibly, if we add extra features to these vectors, i.e. concave features or crossed lines features, the result would be even better.

Zernike moments are better than geometric moments, but are not so good as gradient features. In this paper we mainly focus on normalization methods and as a matter of fact we do not investigate the feature vectors. Maybe methods presented in [7] can be extended and yield to better results.

The best results are achieved using gradient features. The extended GFC vector leads to even better results. The geometric moment invariants and Zernike moments appear slightly worse, but the results are promising. Perhaps combining these feature vectors with other features will lead to better recognition ratios. GF and GFC vectors are most promising. Considering that they are easy to extract and easy to understand they seem to be good choice for future work.

To examine how normalization methods influence recognition rates all feature vectors are extracted from the dataset also without normalization (the column WN in Tab. 6 shows the recognition rate on this vector). Tab. 7 presents relative recognition rates defined as follows:

$$RRR_{ij} = \left(1 - \frac{100 - R_{ij}}{100 - R_i}\right) * 100,$$

where $R_{ij}$ is recognition rate obtained on $i$-th feature vector using $j$-th normalization and $R_i$ is recognition rate obtained on $i$-th feature vector without any normalization.

This measure shows how a normalization method contributes to achieving the optimal recognition rate. For example RRR = 100 means that this normalization method leads to the maximal recognition rate = 100%, RRR = 0 means that the corresponding normalization method brings no advantage to the result and values less than zero mean that the normalization deteriorates the recognition ratio.

**Tab. 7.** Relative recognition rates

| | N0 | N1 | N2 | N3 | N4 | N5 | N6 | N7 | N8 | N9 | N10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| MIs | 15.8 | 20.2 | 29.5 | 28.4 | 30.0 | 26.1 | 27.2 | 29.6 | 29.8 | 27.6 | 27.5 |
| ZM1 | 46.0 | 55.0 | 44.2 | 59.4 | 61.9 | 48.4 | 55.3 | 64.3 | 63.6 | 63.5 | 62.6 |
| ZM2 | 43,3 | 55.7 | 56.5 | 57.9 | 51.2 | 42.7 | 52.4 | 62.1 | 60.9 | 62.0 | 58.7 |
| GF | 41.2 | 42.2 | 44.9 | 63.7 | 56.4 | 60.4 | 58.0 | 77.4 | 74.6 | 77.6 | 76.6 |
| GFC | 55.0 | 50.6 | 62.4 | 68.1 | 64.7 | 67.9 | 60.5 | 82.4 | 75.2 | 78.6 | 80.3 |

## 6. Conclusions and future work

In this paper several normalization methods and five feature vectors are compared on the MNIST database. The recognition results show that the moment normalization functions N7 yield the highest recognition rates. The results obtained using moment normalization functions N10 and N9 are also very good. Generally, the normalization is influential to the recognition performance for both dimension-based and moment-based normalization. It is interesting that preserving the aspect ratio or forcing the aspect ratio to one leads to substantially worse results.

There are five sets of feature vectors tested. The best results are achieved using gradient features, but it can be seen that other feature vectors are not quite useless. The results are promising. Maybe some extension of these feature vectors could lead to results comparable with gradient features. The reported results provide useful insights for selecting a suitable normalization algorithm in developing recognition systems.

There are also interesting results in experiments using different aspect ratios. Testing one class versus all others shows that there is no universal aspect ratio optimal for all classes. Different aspect ratios are optimal for different numerals. For example for numeral 1 the best aspect ratio is 0.4 and

for numeral 0 the best aspect ratio is 0.95. This observation is useless in this experiment, because there must be one classifier for all numerals, but in the future work it can be used for building a multiple classifier solution.

The experiments described in this paper are focused on normalization. The results show that this preprocessing technique has a great impact on the final recognition rate regardless of the feature vector used. The next step is to find even better feature vectors which in conjunction with these normalization techniques will lead to even better recognition rates.

## 7. References


[1] Xu D., Li H.; *Geometric moment invariants,* Pattern Recognition 41, 2008, pp. 240–249.

[2] Liu Ch.L., Nakashima K., Sako H., Fujisava H.; *Handwritten digit recognition: investigation of normalization and feature extraction techniques,* Pattern Recognition 37, 2004, pp. 265–279.

[3] Chang C.-C., Lin C.-J.; *LIBSVM: a library for support vector machines,* software available at: http://www.csie.ntu.edu.tw/~cjlin/libsvm, 2001.

[4] Lauer F., Suen Ch.Y., Bloch G.; *A trainable feature extractor for handwritten digit recognitio*n, Pattern Recognition 40, 2007, pp. 1816–1824.

[5] Zhang W., Tang Y.Y., Xue Y.; *Handwritten Character Recognition Using Combined Gradient and Wavelet Feature*, International Conference on Computational Intelligence and Security, Vol. 1, 2006, pp. 662–667.

[6] Stąpor K.; *Automatic object classification*, Publishing House EXIT, 2005.

[7] Tong X.J., Zeng S., Zhou K., Jiang Q.; *Hand-written numeral recognition based on Zernike moment*, Proceedings of the 2008 ICWAPR, pp. 368–372.

[8] Teow L.-N., Loe K.-F.; *Robust vision-based features and classification schemes for offline handwritten digit recognition*, Pattern Recognition 35(11), 2002, pp. 2355–2364.

[9] Liu H., Ding X.; *Handwritten Character Recognition Using Gradient Feature and Quadratic Classifier with Multiple Discrimination Schemes*, Proceedings of the Eighth ICDAR, 2005, pp. 19–25.

[10] Shi M., Fujisava Y., Wakabayashi T., Kimura F.; *Handwritten Numeral Recognition using gradient and curvature of gray scale image*, Pattern Recognition 35(10), 2002, pp. 2051–2059.

[11] Cristianini N., Scholkopf B.; *Support vector machines and Kernel methods: the new generation of learning machines*, AI Magazine 13(3), 2002, pp. 3–41.

[12] Liu Ch.L., Nakashima K., Sako H., Fujisava H.; *Handwritten digit recognition: benchmarking of state-of-the-art techniques*, Pattern Recognition 36, 2003, pp. 2271–2285.

[13] Shi M., Fujisawa Y., Wakabayashi T., Kimura F.; *Handwritten numeral recognition using gradient and curvature of gray scale image*, Pattern Recognition 35, 2002, pp. 2051–2059.

[14] Scholkopf B., Smola A.J.; *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond,* The MIT Press, 2001.

[15] Cheriet M., Kharma N., Liu Ch.-L., Suen Ch.-Y.; *Character Recognition Systems: A guide for students and practioners,* Wiley-Interscience, 2007.

[16] Suen Ch.Y., Nadal Ch., Legault R., Mai T.A., Lam L.; *Computer Recognition of unconstrained handwritten numeral*, Proceedings of the IEEE, 80, 1992, pp. 1162–1180.

[17] Srikantan G., Lam S.W., SriHari S.N.; *Gradient-based contour encoding for character recognition*, Pattern Recognition 29, 1996, pp. 1147–1160.

[18] Arica N., Yarmna-Vural F.T.; *Optical Character Recognition for Cursive Handwriting*, IEEE Transactions on Pattern Analysis and Machine Intelligence 23, 2002, pp. 801–813.

[19] Liu C.-L., Nakashima K., Sako H., Fujisawa H.; *Aspect Ratio adaptive normalization for handwritten character recognition*, in: Advances in Multimodal Interfaces – ICMI 2000, T. Tan Y. Shi, W. Gao (eds.), Lecture Notes in Computer Science 1948, 2000, pp. 418–425.

[20] Kimura et al.; *Evaluation an synthesis of feature vectors for handwritten numeral recognition*, IEICE Trans. Inform. Systems E79-D(5), 1996, pp. 436–442.

[21] Heutte L., Paquet T., Moreau J.V., Lecourtier Y., Olivier C.; *A structural/ statistical feature based vector for handwritten character recognition*, Pattern Recognition Letters 19(7), 1998, pp. 629–641.

[22] Jain A.K., Duin R.P.W., Mao J.; *Statistical Pattern Recognition: a review*, IEEE Transactions on Pattern Analysis and Machine Intelligence 22(1), 2000, pp. 4–37.

[23] Burges C.J.C.; A *tutorial on support vector machines for pattern recognition*, Knowledge Discovery Data Mining 2(2), 1998, pp. 1–43.

[24] Kimura F., Takashina K., Tsuruoka S., Miyake Y.; *Modified quadratic discriminant functions and the application to Chinese character recognition*, IEEE Trans. Pattern Anal. Mach. Intell. 9(1), 1987, pp. 149–153.

[25] Vapnik V.; *The Nature of Statistical Learning Theory*, Springer, New York 1995.

[26] Kimura F., Shridhar M.; *Handwritten numeral recognition based on multiple algorithms*, Pattern Recognition 24(10), 1991, pp. 969–981.

[27] Liu C.-L., Sako H., Fujisawa H.; *Performance evaluation of pattern classifiers for handwritten character recognition*, International Journal on Document Analysis Recognition 4(3), 2002, pp. 191–204.

[28] Lee D.-S., Srihari S.-N.; *Handprinted digit recognition: a comparison of algorithms*, Proceedings of the Third International Workshop on Frontiers of Handwriting Recognition, Buffalo, New York, 1993, pp. 153–164.

[29] LeCun Y. et al.; *Comparison of learning algorithms for handwritten digit recognition*, in: Proceedings of the International Conference on Artificial Neural Networks, F. Fogelman-Soulie, P. Gallinari (eds.), Nanterre, France 1995, pp. 53–60.

[30] Suen C.-Y., Liu K., Strathy N.W.; *Sorting and recognizing cheques and financial documents*, in: Document Analysis Systems: Theory and Practice, S.-W. Lee, Y. Nakano (eds.), Springer, Berlin 1999, pp. 173–187.

[31] Liu C.-L., Nakagawa M.; *Handwritten numeral recognition using neural networks: improving the accuracy by discriminative training*, Proceedings of the Fifth International Conference on Document Analysis and Recognition, 1999, pp. 257–260.

[32] Naggy G., Tuong N.; *Normalization techniques for handprinted numerals*, Communications of the ACM 13(8), 1970, pp. 475–481.

[33] Franke J.; *Isolated handprinted digit recognition*, in: Handbook of Character Recognition and Document Image Analysis, H. Bunke, P.S.P. Wang (eds.), World Scientific, Singapore 1997, pp. 103–121.

[34] Gader P.D., Khabou M.A.; *Automatic feature generation for handwritten digit recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence 18(12), 1996, pp. 1256–1261.

[35] Cai J.-H., Liu Z.-Q.; *Integration of structural and statistical information for unconstrained handwritten numeral recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence 21(3), 1999, pp. 263–270.

[36] Oh I.-S., Lee J.-S., Suen C.Y.; *Analysis of class separation and combination of class-dependent features for handwriting recognition*, IEEE Transactions on Pattern Analysis and Machine Intelligence 21(10), 1999, pp. 1089–1094.

[37] Mayraz G., Hinton G.E.; *Recognizing handwritten digits using hierarchical products of experts*, IEEE Transactions on Pattern Analysis and Machine Intelligence 24(2), 2002, pp. 189–197.

[38] Dong J.X., Krzyzak A., Suen C.Y.; *A multi-net learning framework for pattern recognition*, Proceedings of the Sixth International Conference on Document Analysis and Recognition, Seattle 2001, pp. 328–332.

[39] Belongie S., Malik J., Puzicha J.; *Shape matching and object recognition using shape contexts*, IEEE Transactions on Pattern Analysis and Machine Intelligence 24(4), 2002, pp. 509–522.

[40] Gonzalez R.C., Woods R.E.; *Digital Image Processing*, 2nd edition, Addison Wesley, 2001.

[41] Hull J.J.; *Document image skew detection: Survey and annotated bibliography,* in: Document Analysis Systems II, J.J. Hull and S.L. Taylor (eds.), World Scientific, Singapore, 1998, pp. 40–64.

[42] Yamaguchi T., Nakano Y., Maruyama M., Miyao H., Hannoi T.; *Digit classification on siggnboard for telephone number recognition*, Proceedings of the 7th International Conference for Document Analysis and Recognition, Edinburgh, Scotland 2003, pp. 359–363.

[43] Zhang T.Y., Suen C.Y.; *A fast parallel algorithm for thinning digital patterns*, Communication of the ACM 27(3), 1984, pp. 236–239.

[44] Favata J.T., Srikantan G., Srihari S.N.; *Handprinted character/digit recognition using a multiple feature/resolution philosophy*, Proceedings of the Fourth International Workshop on Frontiers of Handwriting Recognition, Taipei 1994, pp. 57–66.

[45] de Oliveira Jr. J.J., Veloso L.R., de Carvalho J.M.; *Interpolation/decimation scheme applied to size normalization of characters images*, Proceedings of the 15th International Conference Pattern Recognition, Vol. 2, Barcelona, Spain 2000, pp. 577–580.

[46] Ramteke R.J., Mehrotra S.C.; *Feature Extraction Based on Moment Invariants for Handwriting*, IEEE Conference on Recognition Cybernetics and Intelligent Systems, Issue 7–9, 2006, pp. 1–6.

[47] Cheng D., Yan H.; *Recognition of handwritten digits based on contour information*, Pattern Recognition 31(3), 1998, pp. 235–255.

[48] Tong X.J., Zeng S., Zhou K., Zhao K., Jiang Q.; *Hand-written numeral recognition based on Zernike moment*, Proceedings of the 2008 International Conference on Wavelet Analysis and Pattern Recognition, Vol. 1, 2008, pp. 368–372.

[49] Teague M.R.; *Image analysis via the general theory of moments*, Journal of the Optical Society of America 70(8), 1980, pp. 920–930.

[50] Trier O.D., Jain A.K., Taxt T.; *Feature extraction. Methods for character recognition – a survey,* Pattern Recognition 29, 1996, pp. 641–662.

[51] Kawamura A. et al.; *On-line recognition of freely handwritten Japanese characters using directional feature densities*, Proceedings of the 11th International Conference on Pattern Recognition, Vol. 2, The Hague 1992, pp. 183–186.

[52] Mori S., Suen C.Y., Yamamoto K.; *Historical review of OCR research and development,* Proceedings of IEEE 80(7), 1992, pp. 1029–1053.

[53] Khotanzad A., Hong Y.H.; *Invariant image recognition by Zernike moments*, IEEE Transactions on Pattern Analysis and Machine Intelligence 12(5), 1990, pp. 489–490.

[54] Burges C.J.C.; *A tutorial on support vector machines for pattern recognition*, Knowledge Discovery Data Mining 2(2), 1998, pp. 1–43.

[55] Gudessen A.; *Quantitative Analysis of preprocessing techniques for the recognition of handprinted characters,* Pattern Recognition 8, 1976, pp. 219–227.

[56] Cristianini N., Shawe-Taylor J.; *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods,* Cambridge University Press, 2000.

[57] Labusch K., Barth E., Martinetz T.; *Simple Method for High-Performance Digit Recognition Based on Sparse Coding*, IEEE Transaction on Neural Networks 19(11), 2008, pp. 1985–1989.

[58] Fan R.E., Chen P.H., Lin C.J.; *Working Set Selection Using Second Order Information for Training Support Vector Machines*, Journal of Machine Learning Research 6, 2005, pp. 1889–1918.

[59] Keerthi S.S., Lin C.J.; *Asymptotic behaviors of support vector machines with Gaussian kernel,* Neural Computation 15(7), 2003, pp. 1667–1689.

[60] Kernel machines web site, http://www.kernel-machines.org/.

[61] The MNIST database of handwritten digits, http://yann.lecun.com/exdb/mnist/.

[62] Hsu C.W., Chang C.C., Lin C.J.; *A practical guide to support vector classification,* http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

[63] Mukundan R., RamaKrishnan K.R.; *Fast Computation of Legendre and Zernike Moments*, Pattern Recognition 28(9), 1995, pp. 1433–1442.

[64] Casey R.G.; *Moment normalization of handprinted character,* IBM Journal of Research and Development 14, 1970, pp. 548–557.

[65] Lee S.-W.; *Multilayer cluster neural network for totally unconstrained handwritten numeral recognition*, Neural Networks 8(5), 1995, pp. 783–792.

[66] Abuhaiba I.S.I., Holt M.J.J., Datta S.; *Recognition of off-line handwriting,* Computer Vision and Image Understanding 71, 1998, pp. 19–38.

*Received May 16, 2010*