Sketched Symbol Recognition using Zernike Moments

Heloise Hse and A. Richard Newton
Department of Electrical Engineering and Computer Sciences
University of California at Berkeley
Berkeley, CA 94720, U.S.A.
{hwawen, newton}@eecs.berkeley.edu

Abstract

In this paper, we present an on-line recognition method for hand-sketched symbols. The method is independent of stroke-order, -number, and -direction, as well as invariant to rotation, scaling, and translation of symbols. Zernike moment descriptors are used to represent symbols and three different classification techniques are compared: Support Vector Machines (SVM), Minimum Mean Distance (MMD), and Nearest Neighbor (NN). We have obtained 97% accuracy rate on a dataset consisting of 7,410 sketched symbols using Zernike moment features and a SVM classifier. This method has been implemented in a software recognition package, HHreco [7].

1. Introduction

Sketching is a simple and natural mode of expression. It is especially desirable for conceptual design, both on an individual basis and in a collaborative environment. With a sketch-based user interface, one can have the freedom of sketching on paper and the benefit of an electronic design tool. If a sketch system also includes a recognition capability, sketches can be interpreted and augmented with semantics so that they can be edited easily, efficiently searched, and neatened.

There has been a significant amount of research to date in various aspects of sketch-based user interfaces: interactive design tools [12], studies of gestures [14], software toolkits [6], ink beautification [8], and sketch recognition [19]. In this work, we focus on the recognition of graphic symbols used in common applications. Challenges in sketched symbol recognition lie in the variation and distortion of hand-sketched shapes. Different people may use different stroke-order, -number, and -direction to draw the same shape. Unlike printed symbols, hand-sketched symbols are imprecise in nature such that corners are not always sharp, lines are not perfectly straight, and curves are not necessarily smooth. Furthermore, symbols can be drawn in different sizes and orientation (e.g. the orientation of an arrow depends on its pointing direction), in contrast to handwriting which is often assumed to be written on a baseline in an upright position. A robust recognition system has to account for all of these factors.

The work in on-line sketched symbol recognition can be roughly categorized into statistical [4, 16], structural [2, 15], and rule-based approaches [1, 25]. The structural approach describes a symbol in terms of simpler geometric primitives and represent it with a semantic network [2]. However, mistakes in the segmentation stage can lead to incorrect descriptions of symbols and inexact graph matching in the classification stage is very computationally expensive [17]. Rule-based approaches are often ad-hoc, hard to extend, and not very robust. In this work, we consider a statistical approach to sketched symbol recognition using Zernike moments as features. Zernike moments have been used

in the optical character recognition and image recognition communities with good results [10, 22], but as far as we know, this feature has not been explored for use in on-line symbol recognition. The rotation invariance property of Zernike moments is especially desirable for symbol recognition.

In this work, we evaluate the effectiveness of Zernike moment features for hand-drawn symbol recognition along with three classification methods. Section 2 gives a description of our target symbol set and the test corpus. Symbol preprocessing, feature extraction, and classification methods are presented in Section 3, 4, and 5 respectively. Experiments and results are presented in Section 6.

2. Data acquisition

Since there is no publicly available benchmark for sketched symbols, we have created a test corpus by gathering data from different people. Our target class of application for this work is one that has a bounded set of target symbols from which to select (e.g. a UML diagram editor, a slide drawing program like Microsoft PowerPoint, or an electrical schematic editing tool). The shape set was chosen based upon the applications of interest, commonly used basic shapes, and the geometric properties of shapes (e.g. shapes with lines, shapes with curves, shapes with mixed lines and curves, and shapes with and without self intersections). Of course, other shapes can be added and learned by the system, if desired.

So far, we have gathered data from 19 users. Each user was asked to sketch 30 examples for each of the 13 symbols shown in Figure 1. The data set contains a total of 7,410 examples overall and 570 examples per symbol. The data was collected using the Wacom Graphire2 Pen and Tablet [24].

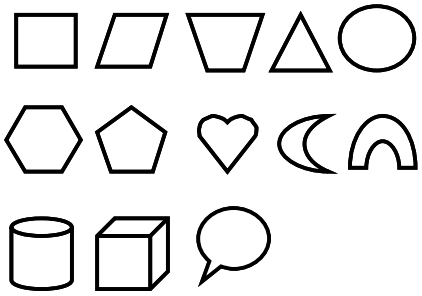

Figure 1. The symbol set.

3. Preprocessing

Zernike moments are not invariant to scale and translation, therefore the symbols are first scaled and translation normalized such that they are of the same dimension and their centroids are positioned at the origin. Each symbol is 100×100 in size after scale normalization.

Each symbol consists of a sequence of strokes. The strokes are approximated and interpolated to produce a more evenly distributed data points for moment calculation. Stroke approximations are used to reduce noise and the number of points enabling faster processing [5]. However, excessive filtering may result in the loss of perceptually salient points (e.g. corners) important to recognition.

We have found that using a threshold value of 1.0 in the stroke filter provides a good balance between point reduction and information retention.

Next, strokes are interpolated to obtain more evenly weighted strokes and hence more consistent moment features. Given a threshold value, *T*, a point is inserted in between two points that are more than *T* distance apart. The procedure continues until the distance between every pair of consecutive points on a stroke is less than *T*. The larger *T* is, potentially fewer points need to be inserted and hence fewer data points in a stroke. This will speed up moment computation but at some point, it may adversely affect the recognition accuracy rate due to insufficient data points. We have experimented with different interpolation thresholds to determine the optimal value. The results are presented in Section 5.

4. Zernike moments

Moment descriptors have been studied for image recognition and computer vision since the 1960s [22]. Teague first introduced the use of Zernike moments to overcome the shortcomings of information redundancy present in the popular geometric moments [13, 21]. Zernike moments are a class of orthogonal moments and have been shown effective in terms of image representation. Zernike moments are rotation invariant and can be easily constructed to an arbitrary order. Although higher order moments carry more fine details of an image, they are also more susceptible to noise. Therefore we have experimented with different orders of Zernike moments to determine the optimal order for our problem.

The Zernike polynomials are a set of complex, orthogonal polynomials defined over the interior of a unit circle $x^2 + y^2 = 1$ [10, 11],

$$V_{nm}(x,y) = V_{nm}(\rho,\theta) = R_{nm}(\rho)e^{jm\theta} \qquad (1)$$

$$R_{nm}(\rho) = \sum_{s=0}^{\frac{n-|m|}{2}} (-1)^s \frac{(n-s)!}{s!\left(\frac{n+|m|}{2}-s\right)!\left(\frac{n-|m|}{2}-s\right)!}\rho^{n-2s} \qquad (2)$$

where n is a non-negative integer, m is an integer such that $n-|m|$ is even and $|m| \le n$, $\rho = \sqrt{x^2+y^2}$, and $\theta = \tan^{-1}\frac{y}{x}$.

Projecting the image function onto the basis set, the Zernike moment of order n with repetition m is:

$$A_{nm} = \frac{n+1}{\pi}\sum_x\sum_y f(x,y)V_{nm}(x,y), \quad x^2+y^2 \le 1 \qquad (3)$$

It has been shown in [11] that the Zernike moments on a rotated image differ from those of the original unrotated image in phase shifts, but not in magnitudes. Therefore $|A_{nm}|$ can be used as a rotation invariant feature of the image function. Since $A_{n,-m} = A_{nm}$, and therefore $|A_{n,-m}| = |A_{nm}|$, we will

use only $|A_{nm}|$ for features. Since $|A_{00}|$ and $|A_{11}|$ are the same for all of the normalized symbols, they will not be used in the feature set. Therefore the extracted features of the order n start from the second order moments up to the n^{th} order moments.

5. Learning and classification

Three different classification techniques have been evaluated. They are Support Vector Machines (SVM) [23], minimum mean distance (MMD), and nearest neighbor (NN) [10]. The classifiers learn from the training set in which every example is represented with a multi-dimensional feature vector composed of extracted Zernike moments.

For the SVM multi-class classifier, we used a radial basis kernel and pair-wise classification [18]. This results in $(N-1)N/2$ binary classifiers where N is the number of class. During classification, all classifiers are evaluated and the test example is classified to the class receiving the maximum number of votes. The training data is scaled to be in the range of [0, 1] in order to avoid numerical problems. The test data is also scaled according to the parameters obtained during the training stage. Since our recognition system is developed in Java, we used the libsvm package which is also written in Java to allow better integration [3].

In the minimum distance classifier, each symbol class, C_k, is represented with the sample means, μ^k, and standard deviations, σ^k learned from the training examples. When a new example is given, it is compared to each symbol class by calculating the normalized Euclidean distance. The normalization is done on each feature to account for the variance in that feature dimension. The example is assigned to class k for which the distance is minimum. The equations are show below:

$$\mu_i^k = \frac{1}{n}\sum_{j=1}^{n} x_{i,j}^k \qquad (4)$$

$$\sigma_i^k = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(x_{i,j}^k - \mu_i^k\right)^2} \qquad (5)$$

$$d(x, C_k) = \sum_{i=1}^{m}\left(\frac{x_i - \mu_i^k}{\sigma_i^k}\right)^2 \qquad (6)$$

where

μ_i^k	=	the mean of the i^{th} feature in class k
σ_i^k	=	the standard deviation of the i^{th} feature in class k
$x_{i,j}^k$	=	the value of the i^{th} feature of example j in class k
$d(x, C_k)$	=	the normalized distance between example x and class k
n	=	the number of examples in class k
m	=	the feature dimension

During training, the nearest neighbor classifier normalizes the feature vectors of the examples in the training classes using the corresponding μ^k, and σ^k. In the classification stage, the classifier extracts features from the test example and computes the normalized Euclidean distance, d, between the

example and every training example. The test example has to be normalized using the parameters of the class under test. The training example of class k, c^k, with the smallest distance to the test example, a, is the nearest neighbor of a. Equation 7 shows the normalization of an example to class k. Equation 8 shows the Euclidean distance between two examples.

$$\hat{a}_i = \frac{a_i - \mu_i^k}{\sigma_i^k} \qquad (7)$$

$$d\left(\hat{a}, \hat{c}^k\right) = \sum_{i=1}^{m} \left(\hat{a}_i - \hat{c}_i^k\right)^2 \qquad (8)$$

The recognition can be made adaptive with each of these classification methods by updating the training parameters with added examples.

6. Experimental results

Two potential methods that a recognition system can be used in an application are that a user can train the recognizer with his or her own data, and that the user can use the pre-trained recognizer which is adaptive. We have designed two sets of experiments based on these two usage scenarios to evaluate the recognition system.

6.1. Experiment 1: writer-dependent test

This experiment evaluates the performance of the recognizer in a writer-dependent situation. A classifier is trained and tested using different parts of an individual's dataset and is evaluated on all users' datasets. K-fold cross validation is used where K is set to 10 [20]. Each symbol is scale and translation normalized, approximated, and interpolated as described in Section 3. Various interpolation values and orders of Zernike moments have been experimented and the results are presented in Table 1–3 and Figure 2.

Table 1. Recognition rates (%) using an SVM classifier.

	Order of Zernike moments							
	3	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Interpolation values **none**	63.6	87.7	90.9	93.0	94.0	94.8	95.2	94.9
10	67.2	89.1	91.4	95.2	96.1	96.9	97.2	97.3
20	67.2	89.1	91.4	95.2	96.1	96.9	97.2	97.3
30	67.2	89.1	91.4	95.2	96.1	96.9	97.2	97.3

Table 2. Recognition rates (%) using an MMD classifier.

	Order of Zernike moments							
	3	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Interpolation values **none**	55.9	81.6	83.5	86.0	86.3	86.4	86.0	85.5
10	61.2	84.2	86.1	89.9	90.4	91.5	90.9	90.8
20	61.2	84.2	86.1	90.0	90.4	91.5	90.9	90.8
30	61.2	84.2	86.1	89.9	90.4	91.5	90.9	90.8

Table 3. Recognition rates (%) using an NN classifier.

	Order of Zernike moments							
	3	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Interpolation values **none**	42.3	79.2	82.4	86.8	87.2	88.0	87.9	87.6
10	47.0	81.4	84.3	90.7	91.2	93.2	92.7	93.0
20	46.8	81.3	84.3	90.7	91.2	93.2	92.7	93.0
30	48.7	81.3	84.5	90.7	91.3	93.2	92.7	93.0

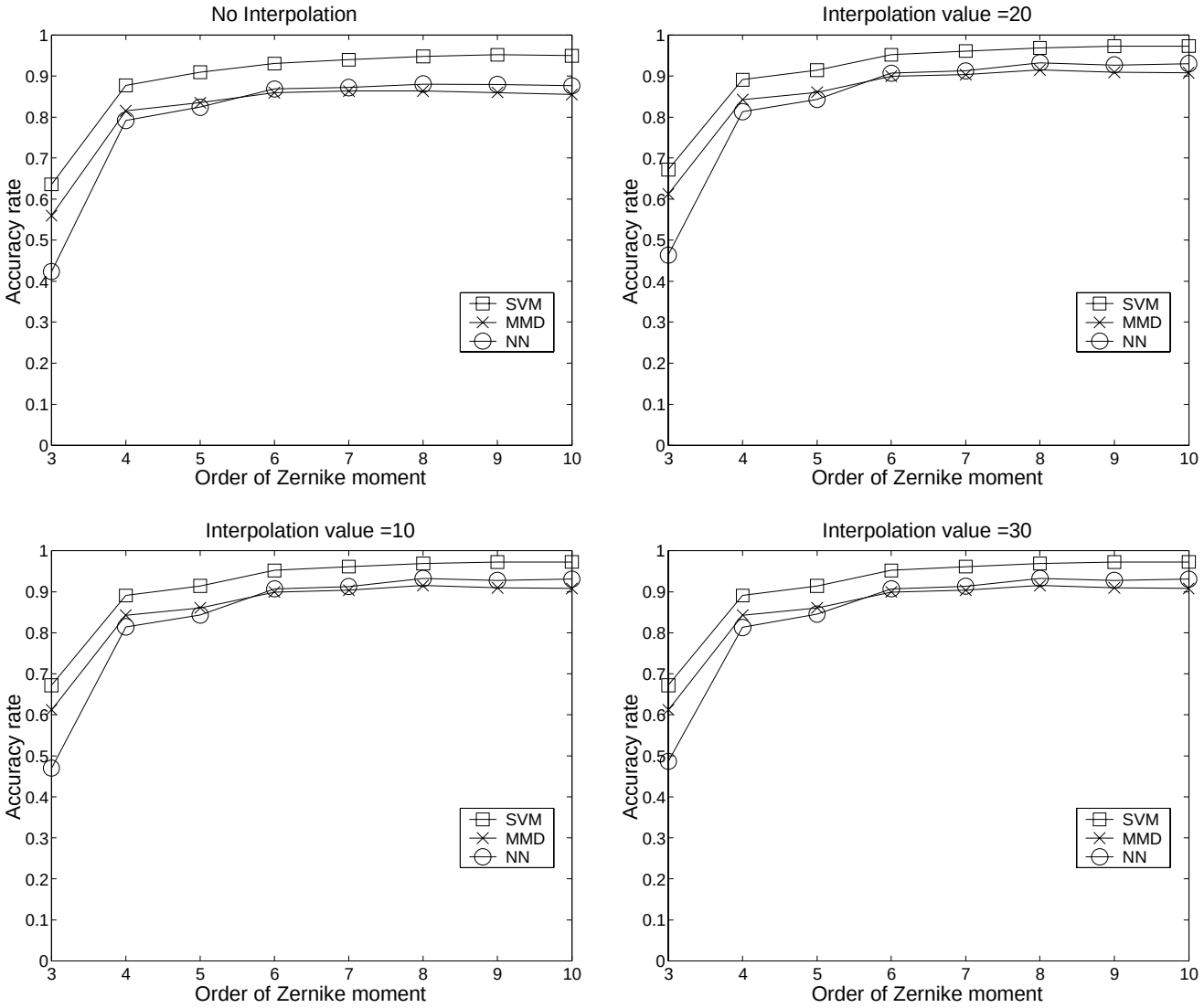

Figure 2. Experiment 1, writer-dependent test.

The result showed SVM is superior compared to the other two classification methods. The accuracy rate is higher when data has been interpolated. However there is very little difference (< 0.2%) in the accuracy rates for interpolation values of 10, 20, and 30. We did not try larger interpolation values for a few reasons: our approximation is very conservative as to not over filter the strokes, so there is little likelihood that neighboring data points would be very far apart, and if they are far apart, they should be interpolated with smaller values anyway to avoid uneven weighting. Accuracy rates start to level off when moment order is 6. Based on the experimental data, order 8 seems to be sufficient for this problem (with interpolation value = 10, the accuracy rates are 96.9% using SVM, 91.5% using MMD, and 93.2% using NN). For a higher order moment, the accuracy rates are slightly higher in the SVM case, but worse in the other two (order = 10, 97.3% using SVM, 90.8% using MMD, and 93.0% using NN).

It is also important to determine the number of examples sufficient to train the recognizer. Since the recognizer is adaptive, it will continue to train itself as more examples are added. However, in this test, we try to determine the number of examples needed to get it started at a practical accuracy rate

(Figure 3). The results of different interpolation values (e.g. 10, 20, 30) are very similar, therefore, we simply show one of the plots below.

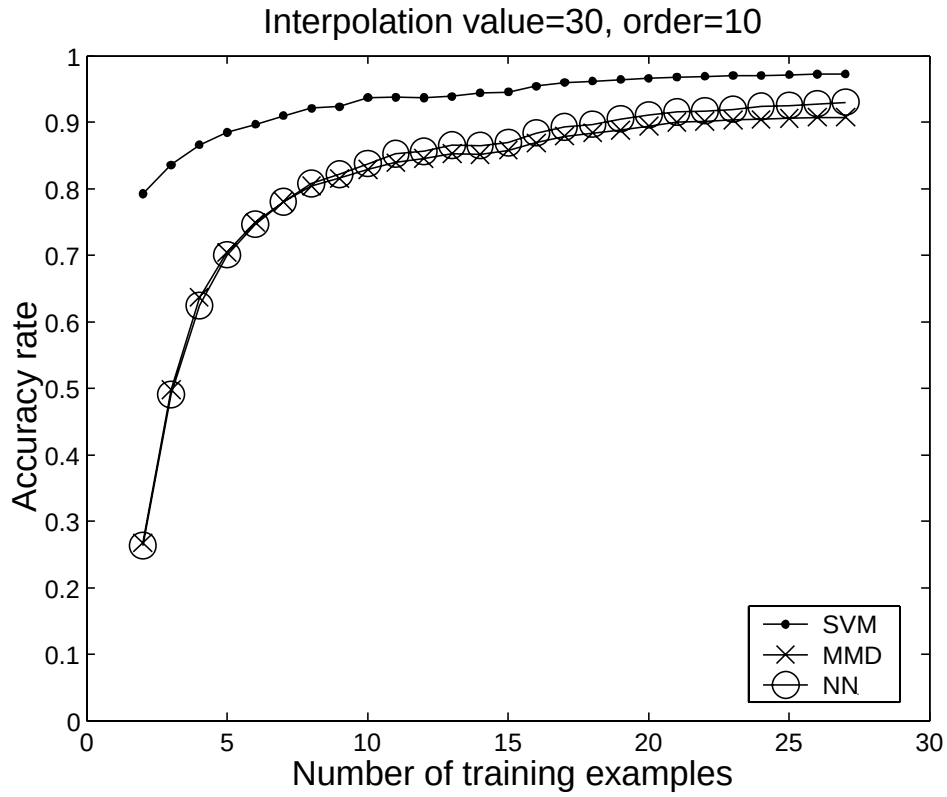

Figure 3. Testing the recognizer on incremental training data.

The accuracy rate converges faster when an SVM classifier is used. For the set of symbols under test, averaged across all users, by providing ≈10 examples per symbol, a user can expect to obtain > 90% accuracy rate using SVM in the writer-dependent case. Though this may seem to be a very small number of training examples for such high accuracy rate using a statistical classifier, we believe it is because there is a great level of consistency in how a user draws shapes. Of course, the more examples, the better it is to train the recognizer.

6.2. Experiment 2: writer-independent test

In this experiment, we are interested in determining how well the pre-trained recognizer works for a new user under different classification methods. *N*–fold cross-validation is used where *N* is the number of users. Each time, a different individual's data set is held out for a test set, and a classifier is trained with all other users' data and then test on the holdout set. For each round, there are ≈ 7,020 symbols for training, and ≈ 390 symbols for testing. Again, various interpolation values and orders of moments have been evaluated along with three different classification methods. The results are shown in Table 4-6 and Figure 4.

Table 4. Recognition rates (%) using an SVM classifier.

	Order of Zernike moments							
Interpolation values	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
none	44.1	77.2	84.1	88.5	89.5	91.0	91.8	92.3
10	63.1	76.7	87.9	94.1	94.4	96.7	96.9	96.2
20	63.1	76.7	87.9	94.1	94.4	96.7	96.9	96.2
30	63.1	76.7	87.9	94.1	94.4	96.7	96.9	96.2

Table 5. Recognition rates (%) using an MMD classifier.

	Order of Zernike moments							
Interpolation values	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
none	30.0	60.5	65.9	70.8	73.3	76.9	75.9	74.1
10	55.4	73.8	78.5	85.4	85.4	92.8	91.5	91.5
20	56.7	75.4	78.5	85.6	85.9	92.8	92.1	91.5
30	56.7	75.4	78.5	85.6	85.9	92.8	92.1	91.5

Table 6. Recognition rates (%) using an NN classifier.

	Order of Zernike moments							
Interpolation values	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
none	8.97	30.5	49.2	72.8	76.2	83.3	81.8	83.8
10	13.3	35.4	60.5	84.6	87.7	94.1	91.3	92.1
20	12.8	32.8	63.6	85.6	88.2	94.4	91.5	92.6
30	12.8	32.8	63.6	85.6	88.2	94.4	91.5	92.6

Again, SVM outperforms the other two classifiers. With a moment order of 6, the accuracy rate starts to level off. We speculate the reason SVM is more robust than the other two methods is due to its discriminative approach which makes use of weaker assumptions on class densities than the generative approach [9]. In the generative approach, such as MMD, if the knowledge of the class densities is reflective of the true data-generation process, then this approach can be more efficient in terms of fewer number of data required. NN is similar to SVM in that it is based on the discriminative models, but it discriminates between every pair of examples instead of every class and the ability to generalize is thus weaker. Also in the case of NN, as the size of the training set increases, the recognition speed will decrease since the number of comparisons needed is proportional to the number of training examples. The experimental data suggests that a moment order of 8 is optimal (with interpolation value = 10, SVM = 96.7 %, MMD = 92.8%, and NN = 94.1%). As evident from the plots, in some cases, the accuracy rates start to decline with higher moment orders.

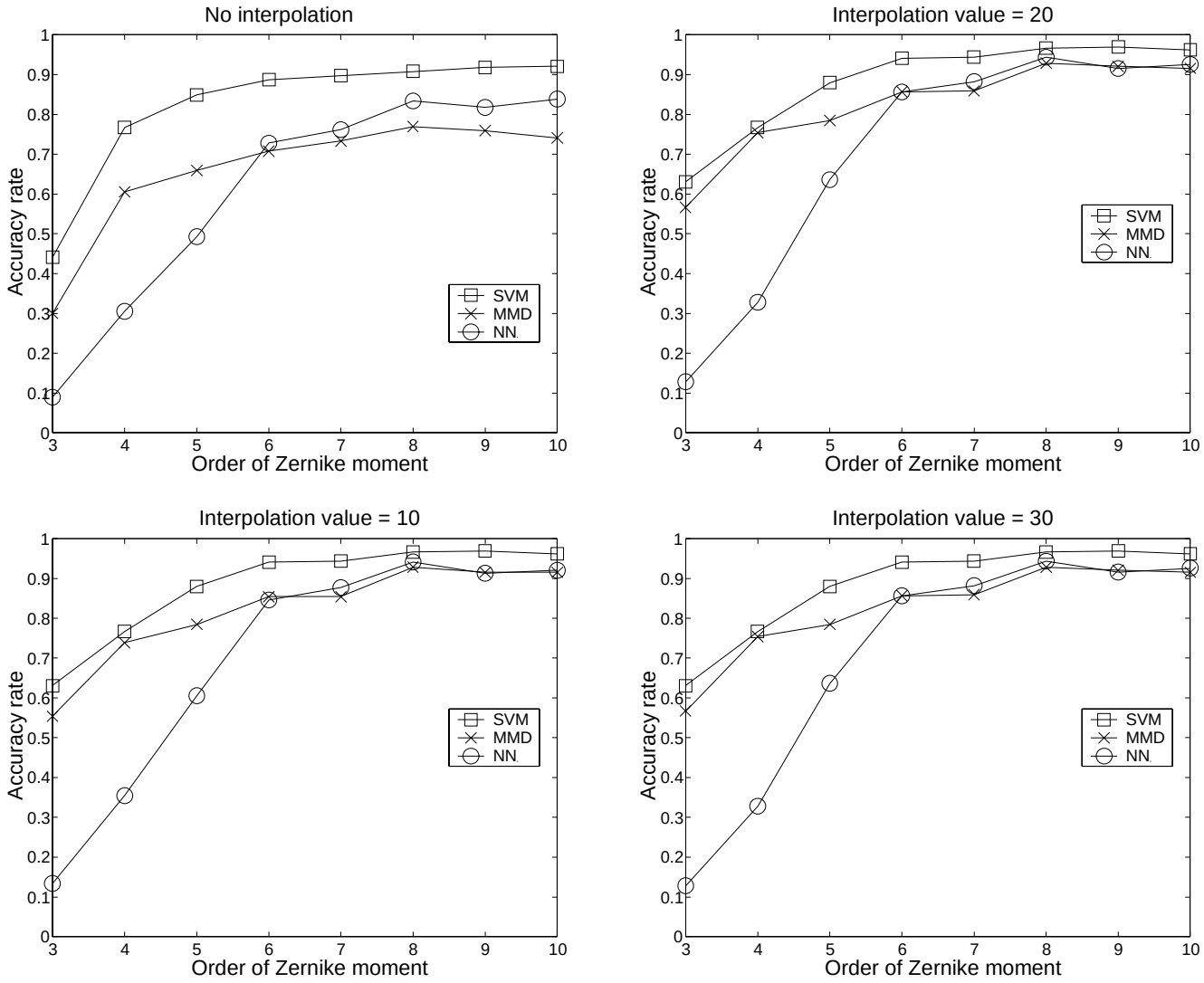

Figure 4. Experiment 2, writer-independent test

7. Conclusion

We have developed an on-line symbol recognition method that is independent of stroke-order, -number, and -direction, as well as invariant to rotation, scaling, and translation. The method is tolerant to shape distortion and adaptive. Using Zernike moments as symbol features, three classification methods, namely SVM, MMD, and NN, have been compared. A recognition accuracy rate of 97% had been obtained using Zernike moments up to order 8 with an SVM classifier.

8. Acknowledgement

This work was supported in part by Microsoft Research Laboratories. Their support is gratefully acknowledged

9. References

[1] Apte, A., Vo, V. and Kimura, T.D., Recognizing Multistroke Geometric Shapes: An Experimental Evaluation. in UIST 1993, (Atlanta GA, 1993), ACM Press, 121-128.

[2] Calhoun, C., Stahovich, T.F., Kurtoglu, T. and Kara, L.B., Recognizing Multi-Stroke Symbols. in 2002 AAAI Spring Symposium on Sketch Understanding, (Palo Alto CA, 2002), AAAI Press, 15-23.

[3] Chang, C. C. and Lin, C. J. http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/.

[4] Fonseca, M.J., Pimentel, C. and Jorge, J.A., CALI: An Online Scribble Recognizer for Calligraphic Interfaces. in 2002 AAAI Spring Symposium on Sketch Understanding, (Palo Alto CA, 2002), AAAI Press, 51-58.

[5] Hanaki, S., Temma, T. and Yoshida, H. An On-line Character Recognition Aimed at a Substitution for a Billing Machine Keyboard. Pattern Recognition, 8. 63-71.

[6] Hong, J.I. and Landay, J.A. SATIN: A Toolkit for Informal Ink-based Applications. CHI Letters: ACM Symposium on UIST, 2 (2). 63-72.

[7] Hse, H. and Newton, A.R. Graphic Symbol Recognition Toolkit (HHreco) Tutorial, Technical Memorandum UCB/ERL M03/50. Electronics Research Lab, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, 2003.

[8] Igarashi, T., Matsuoka, S., Kawachiya, S. and Tanaka, H., Interactive Beautification: a Technique for Rapid Geometric Design. in UIST 1997, (Canada, 1997), 105-114.

[9] Jordan, M.I. An Introduction to Probabilistic Graphical Models, to appear.

[10] Khotanzad, A. and Hong, Y.H. Invariant Image Recognition by Zernike Moments. IEEE Trans. on PAMI, 12 (5). 289-497.

[11] Khotanzad, A. and Hong, Y.H. Rotation Invariant Image Recognition using Features Selected via a Systematic Method. Pattern Recognition, 23 (10). 1089-1101.

[12] Landay, J.A. and Myers, B.A. Sketching Interfaces: Toward More Human Interface Design. IEEE Computer, 34 (3). 56-64.

[13] Lipscomb, J.S. A Trainable Gesture Recognizer. Pattern Recognition, 24 (9). 895-907.

[14] Long, A.C., Landay, J.A. and Rowe, L.A. Visual Similarity of Pen Gestures. CHI Letters: ACM Symposium on UIST, 2 (1). 360-367.

[15] Pavlidis, T. Structural Pattern Recognition. Springer-Verlag Press, Berlin, 1977.

[16] Rubine, D. Specifying Gestures by Example. SIGGRAPH '91, 25 (4). 329-337.

[17] Schalkoff, R. Pattern Recognition: Statistical, Structural and Neural Approaches. John Wiley & Sons, Inc., 1992.

[18] Scholkopf, B. and Smola, A.J. Learning with Kernels. MIT Press, Cambridge, MA, 2002.

[19] Shilman, M., Pasula, H., Russell, S. and Newton, A.R., Statistical Visual Language Models for Ink Parsing. in 2002 AAAI Spring Symposium on Sketch Understanding, (Palo Alto CA, 2002), AAAI Press, 126-132.

[20] Stone, M. Cross-validatory choice and assessment of statistical predictions (with discussion). J. Royal Statist. Soc. B, 36. 111-147.

[21] Teague, M.R. Image Analysis via the General Theory of Moments. Journal of the Optical Society of America, 70 (8). 920-930.

[22] Teh, C. and Chin, R.T. On Image Analysis by the Methods of Moments. IEEE Trans. on PAMI, 10 (4). 496-513.

[23] Vapnik, V. The Nature of Statistical Learning Theory. Springer-Verlag, New York, 1995.

[24] Wacom Technology Co. http://www.wacom.com/.

[25] Zhao, R., On-line Geometry Recognition Using C++, an Object-Oriented Approach. in Proceedings of Tools Europe '92, (Dortmund, Germany, 1992), Prentice Hall, 371-378.