Shape Detection Using Gradient Features for Handwritten Character Recognition[1]

Sameer Singh
School of Computing
University of Plymouth
Plymouth PL4 8AA
United Kingdom

Abstract - *In this paper I shall describe a new method called String Distance Measurement (SDM) for recognizing handwritten characters. The advantage of the technique is that it can be applied in a generic manner to different applications which involve shape recognition and may be successfully modified for individual applications. The technique is based on the measurement of gradient change. The technique is expected to perform better in uncertain and noisy environments compared to the existing methods. The paper describes the technique, and estimates the error rate performing a cross-validation study with neural networks using SDM pattern recognition.*

Keywords - Pattern Recognition, String Distance Measurement (*SDM*), Handwritten Character Recognition, Neural Networks, Cross-Validation, Conjugate Gradient Method.

1. Introduction

In recent years there have been two kinds of attempts towards the development of shape recognition algorithms: *generic* techniques which may be used for recognizing any shape in a dynamic environment; and *specific* techniques which may be used for a limited set of shapes. Generic techniques are more difficult to implement since the amount of noise, nature of data and measurement methods vary significantly across different applications. However their prime advantage is that they can be applied in a range of situations and in various environments. Since the application data differs across applications, generic techniques cannot rely on feature databases which can only hold finite templates. Rather they utilise computational methods such as neural networks which can learn shapes from the available data. Since generic techniques do not incorporate template matching algorithms, the recognition rates are in most cases lower than those obtained by using specific methods.

S. Singh. Shape Detection Using Gradient Features for Handwritten Character Recognition, *Proc. 13th International Conference on Pattern Recognition (ICPR'96),* Vienna, Austria, vol. III, pp. 145-149 (25-29 August, 1996).

Specific methods of shape recognition are developed especially for specific applications. As an example, high quality character recognition systems are required for the market viability of OCR systems. Such systems need very high recognition rates and work with a fixed amount of characters. It is therefore possible to devise feature vectors for these shapes which may be stored in a database for comparison with the actual shape under consideration. The most common method of determining the class of a new shape is to compute feature vector differences between the new shape and the stored character templates. If the mean squared difference between the feature vector of stored template and the new image scanned is below the threshold limit set, the stored template is chosen as representing the shape under consideration.

Character recognition systems have become popular over the last decade. Recent developments in reducing the cost of hardware and fast processing have significantly enhanced their marketability. In accordance with their widespread need, researchers have applied sophisticated recognition techniques including Quadratic Discriminant Analysis for Arabic characters [1], Hidden Markov Models for English words [2], Hough Transform techniques for Chinese characters [3], Bayesian Classifiers for printed characters [6], Prototype Matching for multifont characters [9] and Local Affine Transformation for handwritten numerals [12].

In this paper I shall propose a generic method of shape analysis - *String Distance Measurement (SDM).* The technique is supposed to be generic in the sense that it can be applied to a variety of applications and improved in a number of ways to suit particular applications. The technique is significantly different than conventional pattern matching techniques and *SDM* can only be used by converting a gray scale image into a binary image. In order to be generic, there were two important objectives while formulating the technique. Firstly, *SDM* features have to be independent of the specific aspects of individual shapes such as thickness of line, size of character/shape, etc. Secondly, the technique should be, to a moderate extent, resistant to the dynamic environmental influences during image acquisition such as improper lighting, uncertainty and impreciseness in data, low resolution, and other factors which vary across applications. In this paper I shall describe the *SDM* methodology, verify the usefulness of the technique by presenting the results of a cross-validation study by using a Neural Network for recognizing a set of alphabets, and discuss how the results can be further improved.

2. Quantifying Gradient Change in Images

The majority of the generic shape recognition methods rely on using the topological features of an image. *SDM* quantifies the property of gradient change as the shape changes. The 16 level gray scale image obtained (in this study) is initially converted into a binary image by assigning every pixel value equal to or greater than 128 a value 1 and all others a value 0. The source image is skeletonized and divided into nine segments containing equal number of pixels. For each segment, its *SDM* feature is calculated. The set of rules employed can be summarized below.

Rule 1. The whole process proceeds in a horizontal manner, row by row. At any one time, two successive rows are under consideration.

Rule 2. The process continues unless all rows of the image pixels have been exhausted. Start with the first row.

Rule 3. Lets call the black pixel found in row *i* as B_i .

Rule 4. Find the distance between B_i and B_{i+1} . Lets say the distance is S_i .

Rule 5. Sum up all the distances for a total of *k* rows, $SD = \sum_{i=1}^{k} S_i$.
SD is the String Distance for the whole segment.

This may be shown below with two examples illustrating positive and negative *SD*s.

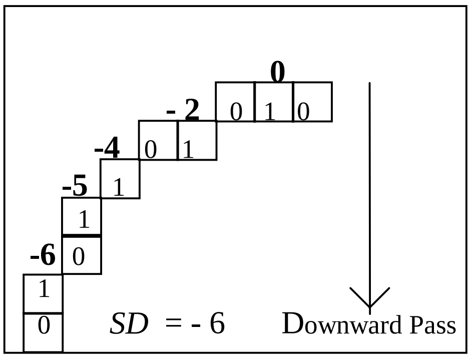


Figure 1. *SD* calculation. (*Negative Values*)

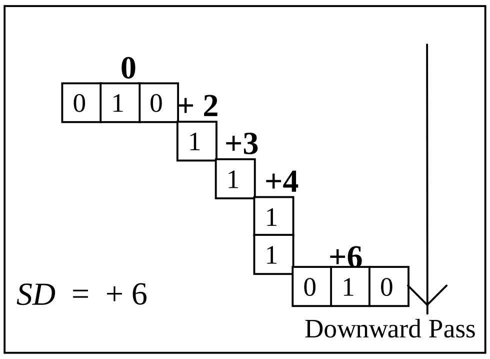


Figure 2. SD Calculation. (*Positive Values*)

SD represents the change in gradient as we traverse in downward direction which is summed up by considering any two adjacent rows at any one time. *SD*s are calculated for each of the nine segments. The overall procedure is described in the next section.

From the above figures it is evident that *SD*s may have a zero value if the Figure 1 & 2 shapes coexist in the same segment. Hence multidirectional lines will contribute differently (positive and negative values) to the summed measurement. Also single horizontal and vertical lines will have zero *SD* values. This may be shown below.

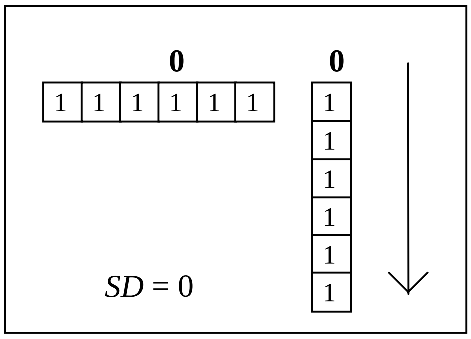


Figure 3. SD Computation (*Zero Values*)

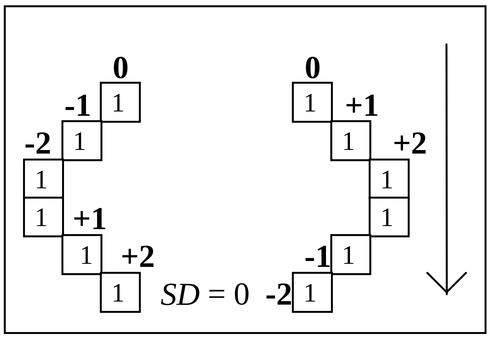


Figure 4. SD Computation (*Zero Values*)

The *SD* computation can be summarized as: (*when looking from the right hand side*) Positive for convex shapes, Negative for concave shapes, and Zero for shapes which are symmetrical around the *x* or *y* axis. However it is needed to include the extent of horizontal and vertical lines which are not properly measured by *SD* alone. Another parameter called String Line Measurement (*SLIM*) is therefore introduced. *SLIM* has a value 1 if a horizontal or vertical line is detected, otherwise it is 0. This *SLIM* value must be scaled to the SD measurement before it can be added to it. Also *SLIM* values must be modified to represent the contribution of longer lines as greater to the overall SDM value. This can be achieved by taking into consideration the total number of black pixels in the straight horizontal or vertical lines with respect to the total number of black pixels in the segment. The final equation used for the string distance measurement of segment *i* with a total of *n* V/H (*vertical or horizontal*) segments is shown below.

$$SDM_i = \alpha\, factor_i + \sum_{j=1}^{n} \beta\, factor_j \quad \text{where}$$

$$\alpha\, factor_i = SD_i$$

$$\beta_j\ factor_i = SD_i * (SLIM_j * \text{Black Pixels within V/H segment}_j\,) / \text{Black Count}_i\,)$$

Here *Black Count* represents the total number of black pixels within the individual segment (one of the nine) under consideration. Horizontal and Vertical scans were separately needed for each segment for identifying horizontal and vertical lines. If there is more than one black pixel in a row or a column, it indicates the presence of a horizontal or vertical line segment. Longer horizontal or vertical segments yield a higher value for the β *factor* and increase the overall value of *SDM*.

3. SDM Analysis

The overall analysis can be divided into three stages: image acquisition and pre-processing, feature extraction, and neural network analysis. In all 25 writers were asked to write alphabets *a* to *z* (both uppercase and lowercase) inside a set of 30mm x 30mm boxes printed on paper. It was not necessary that the image should touch the box boundary as it was clipped later on. The image was scanned with an EPSON GT-8000 scanner which gave 128x128 pixel resolution in the PBM format. The image was first skeletonized. The skeleton was obtained by making use of a conditional erosion algorithm which eroded the original image with four successive passes: from left, right, up and down, using the condition: " erode if (next pixel is black) and (one or more surrounding pixels is black)". The process continued until the image could no longer be eroded. The resultant image was clipped so that the edges of the image were confined within a fixed boundary. The image was then segmented in 9 parts, each segment with its own binary address. The features (*SDM* values) were extracted for each segment and the complete image was described with a nine component vector **S** = (SDM_1, SDM_2, ... SDM_9). In practice although for two different images, a few of the segments may yield identical SDM values, it would be rare if all the nine values were same for completely different shapes. The patterns obtained were used as inputs to the neural network for recognizing different characters (to be described in next section). The overall process can be shown with the following flowchart.

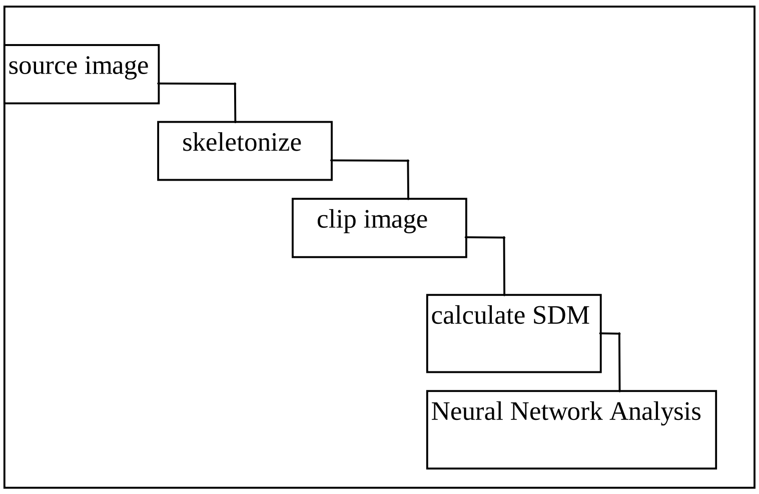
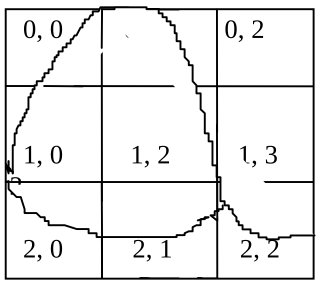


Figure 5. *Flowchart of the Methodology*

The skeletonized image can be shown below.



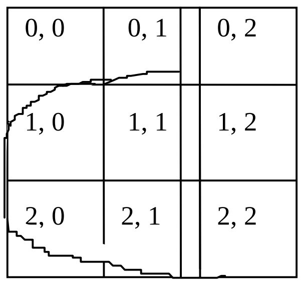


Figure 6. *Skeleton Image of* " a" Figure 7. *Skeleton Image of* " d".

An image processing software was developed which segments the image and analyzes the partitioned segments individually. It can be seen from the above figures that *SDM* values for different segments will be vary across different images, e.g. for the first figure segment (0, 0) it has a negative value, but it is nearly zero for the same segment in the next figure.

4. Neural Network Analysis

In order to evaluate the success of the SDM technique for shape recognition, I decided to use a smaller set of the total number of images scanned. Finally I settled on recognizing images of characters from “ a” to “ e”. This set was chosen since the characters have much more similarity than say characters such as “ a” and “ z” which could be easier to discriminate. Cross-Validation technique was used for error estimation using a 3 layer network (an input layer, a hidden layer, an output layer). The details are presented next.

A. Cross-Validation

There are several reasons for validating learning systems, [4]. The validity of a learning system can be evaluated for these aspects: *performance* - can it provide satisfactory solutions; *generality* -can it applied to many distinct problem domains; *stability* - does it retain previously learnt knowledge; *efficiency* - speed and computational complexity; quality ; *consistency* - will the network perform consistently with similar data; *incremental learning* - will it continue to assimilate new data; and *sensitivity* - to system parameters and training data.

There are several techniques for validating learning systems and error estimation in statistics and pattern recognition such as holdout, leave one out, cross-validation, and bootstrapping. I have decided to make use of cross-validation since it works well will both small and large data-sets. The cross-validation process can be described as “ *K*-fold cross-validation (Stone 1974) repeats *k* times for a sample set randomly divided into *k* disjoint subsets, each time leaving one out for testing and the others for training”, [4]. The main reason for performing a cross-validation exercise is to show the usefulness of *SDM* as a viable recognition technique and to show that that the estimated error rate is the test error rather than the training rate error. The value of *k*, the number of folds, depends on the sample size. Having a very small training set can give poor learning results and very small test sets can give erroneous estimates of the error rate. There are a total of 125 patterns for 25 different writers. I divided the data-set of 125 patterns (25 patterns for each alphabet *a* to *e*) into disjoint training and testing sets to perform a *5*-fold cross-validation. Hence at any one time, there were 100 patterns used for training and 25 patterns used for testing.

B. Neural Network Architecture

It is necessary to have the optimal neural network architecture which does not either over-generalize or only learns the example patterns without generalization. When recognizing handwritten characters, multilayer perceptrons with a large number of neurons are generally used, e.g. [5] have made use of a 4 layer network with 4918 neurons. It is not unusual for some architectures to have a far greater number of layers with millions of connections and thousands of neurons. This increase is driven by the non-linearity in data and an increase in network size makes it excessively difficult to have smaller learning periods. Any real time application can only make use of techniques which can perform quickly, especially if the training is to be performed on-line.

I have made use of a 9x9x5 architecture (9 nodes in the input layer, 9 nodes in the hidden layer and 5 nodes in the output layer). Altogether there are 23 neurons and 126 connections. The training was therefore fast and took less than 500 iterations to complete.

C. Training and Testing

The data consisted of *SDM* values which ranged between approximately -35 and +35. The data was normalized and all of the resulting values were within the [0, 1] range. The network was trained with the normalized data using conjugate-gradient (*CG*) method of training. This method was preferred to the steepest-descent method, since *CG* takes into account the non-linearity of the surface, [7]. It is an efficient method in the sense that it is fast, it does not need the storage of any matrices, and only three vectors have to be stored. The CG procedure does not ask the user to specify any parameters such as momentum or learning rate. CG is a quadratically convergent method based on the concept of conjugate directions. I have made the use of Rudi's Conjugate Gradient method which is a slightly improved version of the conventional CG algorithm and often gives faster convergence (see [7] and [11] for a mathematical description of the method and [8] for its implementation in C programming language).

In every fold there were 100 patterns for training and 25 patterns for testing. The overall procedure was performed by making use of the *Xerion* neural network software developed by van-Camp (see [11]). The software enables the user to graphically visualize the activations, connections, learning methods, error rate, etc. in an object-oriented environment.

D. Results

I shall present the results in terms of the *misclassified* nodes and patterns during testing in each fold. In order to do this, I must clarify what is meant by the terms " *misclassified* nodes" and " *misclassified* patterns" before they are used. By the term " nodes" I shall exclusively refer to output nodes in this section. A node is said to be misclassified if it is classified as something different than the desired target. The threshold limit used is 0.5. As an example, if the desired output is (1 0 0 0 0) and the actual output is (0.2 0 0 0 0.7), there are two misclassified output nodes: *node 1* since the actual output is below 0.5 when it should have been 1.0; and *node 5* whose output is 0.7, above 0.5 limit when the output should have been 0.0. For each fold, the total number of misclassified nodes is recorded during testing. In every test fold, there are in all 125 nodes and 25 patterns which have to be checked for misclassifications.

Pattern misclassifications can be recorded in many ways. One way is to tag a test-pattern as *misclassified* when more than half of the output nodes have been misclassified for that pattern during testing, i.e. in the current problem with a total of 5 output nodes if 3 or more output nodes are misclassified. Since there can be misclassified nodes even in correctly classified patterns (using this method), this may confuse the results and portray a higher recognition rate than what is actually achieved. This method will not be therefore used here. I have decided to represent the results using two separate criteria for *misclassified patterns*. These are:

Criteria I. If the desired result is not obtained, e.g. if the target output is (1 0 0 0 0) and the actual output is (0 1 0 0 0).

Criteria II. i) The desired result is obtained but in competition with another unwanted result, e.g. if the target output is (1 0 0 0 0) and the actual output is (1 1 0 0 0).

ii) The network fails to make a decision (this is rare), e.g. the target output is (1 0 0 0 0) and the actual output is (0 0 0 0 0).

I shall present my results using the above two criteria.

FOLD	*Misclassified Nodes*	*Misclassified Patterns (Criteria I)*	*Misclassified Patterns (Criteria II)*
1	15	8	6
2	9	6	3
3	16	10	6
4	8	4	4
5	15	8	6
TOTAL	63	36	25
Recognition Rate	89.92%	71.2%	80%

Table 1. Pattern Recognition Results

It can be clearly seen that the *SDM* technique gives good results though I have used a small data-set. The technique can be applied to a range of character recognition problems and will hopefully yield very good results when applied to printed characters or shapes.

5. Conclusion

SDM technique can be improved in a number of ways. There is certainly room for further investigation. It may be more advantageous to separately store the positive and negative *SDM* values for each segment. I believe that the above results could be improved in a number of ways. Some of the suggestions would be:

i) Using a larger data set.

ii) Investigating the addition of more hidden layers, although this may not necessarily give a better discriminatory power, [13].

iii) Increasing the number of segments from 9 to 16 to give a larger feature vector.

iv) Modifications to the *SDM* equation.

The SDM approach can be expected to perform better than other existing methods in the presence of noise. This can be shown below.

87 (91)	97 (101)	168 (176)
23 (24)	203 (213)	100 (105)
233 (244)	33 (34)	89 (93)
45 (47)	177 (185)	14 (15)

Figure 8. *A Gray Scale Image*
Values within ( ) are distorted by a noise level of 5%.

0	0	1
0	1	0
1	0	0
0	1	0

Figure 9. *Corresponding Binary Image*.
SDM = - 2 + 1 = -1. No effect of noise.

The proposed approach seems promising with the results obtained. It could be easily applied for recognizing complex shapes. It is not impossible to extend the use of this technique to gray scale images. However gray scale image processing may be more susceptible to noise and uncertainty in data than when dealing with binary images and it would be inappropriate to speculate about probable success without actual results. In conclusion, I hope that this technique will be used favourably in other applications. I have illustrated the use of the proposed technique for handwritten character recognition. The scope is however not limited to character recognition only. It is expected that the technique will give better results than existing shape recognition systems in a number of industrial environments where the image acquisition is susceptible to a number of practical problems such as lighting, resolution, noise, etc. We must welcome any such applications and necessary modifications to the basic proposal of this paper.

REFERENCES

[1] Al-Yousefi, H. and Udpa, S. S. (1992). Recognition of Arabic Characters. *IEEE Transactions on Pattern Analysis*, vol. **14**, no. 8, 853-857.

[2] Chen, M., Kundu, A., and Zhou, J. (1994). Off-Line Handwritten Word Recognition Using a Hidden Markov Model Type Stochastic Network. *IEEE Transactions on Pattern Analysis*, vol. **16**, no. 5, 481-497.

[3] Cheng, F., Hsu, W. and Chen, M. (1989). Recognition of Handwritten Chinese Characters by Modified Hough Transform Techniques. *IEEE Transactions on Pattern Analysis*, vol. **11**, no. 4, 429 - 439.

[4] Fu, L. (1994). Neural Networks in Computer Intelligence (pp. 338-340). Singapore: McGraw-Hill International Editions.

[5] Hussain, B. and Kabuka, M. R. (1994). A Novel Feature Recognition Neural Network and its Application to Character Recognition. *IEEE Transactions on Pattern Analysis*, vol. **16**, no. 1, 98-106.

[6] Kahan, S., Pavlidis, T. and Baird, H. S. (1987). On the Recognition of Printed Characters of Any Font and Size. *IEEE Transactions on Pattern Analysis*, vol**. PAMI-9**, no. 2, 274-289.

[7] McMurtry, G. J. (1994) Adaptive Optimization Procedures. In Jerry M. [8] Mendel (Ed.), *A Prelude to Neural Networks: Adaptive and Learning Systems* (pp. 254-257). New Jersey: Prentice Hall.

[8] Press, W., Flannery, S. T. and Vetterling, W. (1988). Numerical Recipes in C. Cambridge University Press.

[9] Rocha, J. and Pavlidis (1994). A Shape Analysis Model with Applications to a Character Recognition System. *IEEE Transactions on Pattern Analysis*, vol. **16**, no. 4, 393-405.

[10] Stone, M. (1974). Cross-Validatory Choice and Assessment of Statistical Predictions. *Journal of the Royal Statistical Society*, **36**, 111-147.

[11] van-Camp, D. (1992). Xerion Manual. University of Toronto, Department of Computer Science.

[12] Wakahara, T. (1994). Shape Matching Using LAT and its Application to Handwritten Numeral Recognition. *IEEE Transactions on Pattern Analysis*, vol. **16**, no. 6, 618-630.

[13] Weiss, S. M. and Kulikowski, C. A. (1991). Computer Systems that Learn. California: Morgan Kauffman Publishers.