Weighted Elastic Matching Method for Recognition of Handwritten Numerals

Patrice Scattolin
Visual Edge Software
3950 Cote Vertu St-Laurent, Quebec, H4R 1V4, Canada
patrice@vedge.com
patrice@cenparmi.concordia.ca

Adam Krzyzak
Centre for Pattern Recognition and Machine Intelligence GM-606
Concordia University
1455 de Maisonneuve Blvd. West
Montreal, Quebec H3G 1M8, Canada
krzyzak@cs.concordia.ca

Abstract

Elastic matching has been used for the recognition of handwritten characters for two decades in writer-dependent systems using tablet data. We attempt to use elastic matching in the multi-writer environment for recognition of handwritten numerals with tablet data. We modified the elastic matching algorithm by adding weights to each point of the models. These weights allow portions of the characters that better discriminate between classes to take on more importance. In so doing confusing classes are better separated, increasing the recognition rate and reducing the model base size. We do this using a formulation closely related to elastic matching that allows us to add weights without impairing the point tracking capacity of the algorithm.

Keywords: handwriting recognition, elastic matching, on-line data, dynamic programming

1 Introduction

Character recognition is primarily an applied research area, dedicated to solving the problems presented by one specific application: recognizing handwriting in order to convert it into a machine readable format. The result is then used as input to more conventional computer programs. This is echoed in the following quote from Jean Ward [12]:

> *An open question now, after over 30 years of published handwriting recognition research and development, is exactly what useful purpose our work will serve. There are very few, if any, successful applications using handwriting recognition.*

One is forced to admit that, apart from recognition systems, the work done in this field has found little application outside of the confines of the given problem. With this in mind, we looked at the broad requirements of the various character recognition applications. Two parameters stand out, *writer dependence* and *reliability/recognition rate*. A writer-dependent system is one where the performance may be tied to a particular writer, usually performance needs to be high for a given writer with a given writing style, independent of the performance achieved with other writers. Pen based interfaces, benefit from this approach when a system can adapt to a particular writing style.

Attempting recognition of a pattern can have three different outcomes. *Recognition* occurs when the label associated with the pattern matches the identity of the template found in the database. A *substitution* occurs when the associated label differs from the one in the database. Finally a *rejection* occurs when the algorithm cannot associate any label to the unknown. The recognition rate and reliability rate are defined as follows:

App.	Writer Indep.	Performance
Pen Interface	Low	High Reco
Address Rec	High	High Reco
Cheque Pro	High	High Reliability

Table 1: Application Requirements

$$RecRate = \frac{R}{sample} \quad (1)$$

$$Reliability = \frac{R}{(R+S)} \quad (2)$$

where:
R is the number of patterns recognized
S is the number of substitutions
C is the number of confusions

To have both high is ideal but a trade-off can be made in favour of either, often dependent on the cost associated with an error as illustrated by Table 1.

The literature often reports that recognition rates are higher with the tablet data than with the scanned data. The reason being that, with the tablet data, there is more information present and it increases the recognition rate. The characters are made of a continuous sequence of points with a single, unambiguous, predecessor and successors even at junction points. Unfortunately, this is not the case for scanned data. However the increase in performance, although a well established fact, is rather suspicious since the base of reference is the human interpretation of the data. Humans make no use of the dynamic information, since all that is interpreted is the static image of the tablet data. So if the tablet data presents more information, it appears that the dynamic nature, pen speed and acceleration, give little information since humans do not need it to achieve even higher recognition rates. It is then believed that the extra information is not present in the form of speed or acceleration, but in the form of connectivity. Because connectivity can be recovered from scanned data, it is believed that a recognition method used for the tablet data can also be well adapted to the scanned data, while hopefully keeping the advantages of the tablet data recognition methods.

Because of the range of possible performance trade-offs, we propose to modify the elastic matching algorithm, a technique that works well for writer-dependent systems and improve on it to make it a viable writer-independent technique. The result is a system that is fully trainable but controllable. Hopefully the system can answer the needs of both types of applications. It can either be trained to achieve a high recognition rate with a single user, or a high reliability, with a lower recognition rate, for a writer-independent system. This is in contrast with expert system based character recognizers that achieve high rates but are expensive to develop or to modify [8, 7].

We decided on using elastic matching as it was successfully used in the past for single-writer recognition tasks. In addition, it can be adapted to scanned data while, hopefully, keeping the performance advantages associated with tablet data recognition methods. We modified the traditional formula used for elastic matching to allow us to add weights to each point of the models. These weights allow different portions of a given character to take on more importance if they are good discriminators between the correct and the error classes. We present the results of the experiments made with this enhanced formula using a tablet data database collected locally.

2 Preprocessing

Preprocessing is often performed on the tablet data to remove various errors that can occur during digitization [9, 13]. The most frequent problem being spurious points, caused by digitizing tablet errors. To remedy these problems, and remain consistent with the previous work, we implemented preprocessing in three step as illustrated by Figure 1.

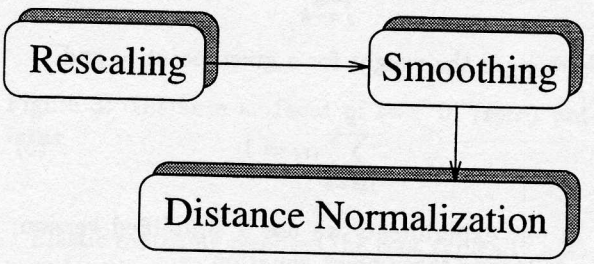

Figure 1: Sequence of generic preprocessing steps.

2.1 Data

The data used was collected during two different sessions for a total of 3300 numerals with 33 different writers [3, 7]. The resulting database was then divided into a training set and a test set of equal size. Since the data from each writer is present in both sets, this makes it a multi-writer system.

2.2 Rescaling

Elastic matching is based on dynamic programming and it minimizes a distance function. This distance function is partly based on the Euclidean distance measurement function. Euclidean distance is sensitive to scale and shift. We must normalize unknowns in some way to prevent introducing a bias. Normalization is performed by applying the following transform $c(x, y)$

$$c = max(100/h, 100/w) \tag{3}$$

Where h and w are the height and width of a pattern in pixels. Such normalization preserves the shape of the pattern while bringing the unknowns generally into a common size, with the largest of either axis 100 pixels in size.

2.3 Smoothing

Aside from the Euclidean distance, angle and curvature measurements are also taken. In order to make these consistent, we must ensure that the curvature is continuous with as few spurious direction changes as possible.

To do so we apply a local smoothing operation on the 2D curve. Such methods are generically defined as weighted averaging of points. For a point p_i on a given 2D curve the resulting smoothed point is p'_i and can be expressed as follows:

$$p'_i = \sum_{j=-k}^{k} \alpha_j p_{i+j} \tag{4}$$

where α_j is the weight for a given point p_i and

$$\sum_{j=-k}^{k} \alpha_j = 1 \tag{5}$$

In our experiments we use a simplified version; the following three point equation:

$$p'_i = \frac{p_{i-1} + p_i + p_{i+1}}{3} \tag{6}$$

It may be applied over several iterations but, experimental results show that the optimal recognition level in our case is achieved with $n = 1$. It is worth mentioning that there are several theoretical criteria to determine what is optimal smoothing. Each criteria results in a slightly different optimal solution. Still, our experimental results are in general agreement with theoretical results of the problem [2].

2.4 Point Distance Normalization

The final preprocessing step is to normalize the distance between points [10, 11]. We take the first point of a point sequence and then remove all the points that are at a Euclidean distance smaller than a given threshold. Because this occurs after rescaling, it ensures that the points are relatively evenly spaced regardless of the size of the original pattern.

Normalization removes the acceleration information present in the data. To find out if acceleration is useful we performed recognition experiments keeping the recognition parameters fixed, varying only the distance normalization threshold. We found that the recognition results improved when the threshold was increased to a distance of 6 pixels between points for normalized patterns. A greater distance between pixels results in a progressive decrease in the recognition rate. If the acceleration information was important, we would expect that the best recognition results would be obtained with a minimum distance of 1 or 2 pixels and then steadily get worse. This is it not what happens. We conclude that acceleration is not useful in this case. This is in stark contrast with the signature verification problem where the dynamic information, hidden from the forger, gives valuable information [4, 5]. That acceleration is not useful is not very surprising considering that recognition by humans is performed on a static representation of the pattern containing no acceleration information.

3 Elastic Matching

Elastic matching is an algorithm derived from the *dynamic programming* technique used for the *string matching* problems. String matching is a process by which one attempts to evaluate the degree of similarity between two sequences by using a cost function associated with elementary operations such as deletion, insertion and substitution.

We obtain a distance from the sum of the costs associated with the operators employed. The goal of minimizing the cost is achieved with the dynamic programming algorithm and is described by Krushkal [1].

3.1 Definition Of Elastic Matching

Elastic matching employs the same strategy to measure dissimilarity between an unknown and known models. A pattern is an ordered sequence of (x, y)-coordinate points forming a string. Elastic matching then measures an unknown string against a set

of reference strings to obtain a distance. The obtained distance is used by the nearest neighbour classifier to classify the unknowns.

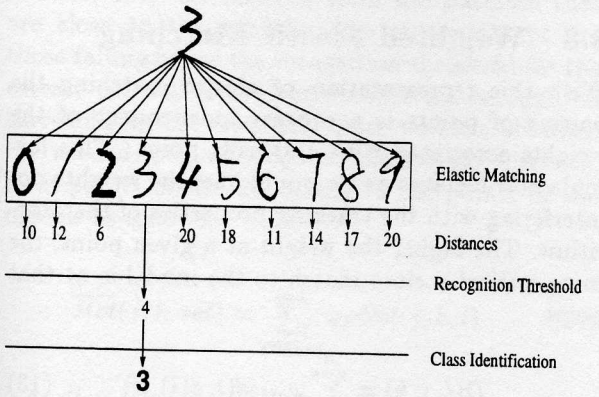

Figure 2: Recognition phase

A frequently used formulation of the elastic matching is given by Tappert [9]. The distance between an unknown sequence of i points and a given model k of j points is expressed as follows:

$$D(i,j;k) = d(i,j;k) + \begin{cases} min \begin{cases} D(i-1,j;k) \\ D(i-1,j-1;k) \\ D(i-1,j-2;k) \end{cases} & j > 2 \\ min \begin{cases} D(i-1,j;k) \\ D(i-1,j-1;k) \end{cases} & j = 2 \\ min \{ D(i-1,j;k) \} & j = 1 \end{cases} \quad (7)$$

where $d(i,j;k)$ is any distance function between two points. We use a linear combination of the Euclidian distance, angle of elevation and curvature:

$$d(i,j;k) = (x_i - x_{j,k})^2 + (y_i - y_{j,k})^2 + c\,|\phi_i - \phi_{j,k}| + b\,|\chi_i - \chi_{j,k}| \quad (8)$$

where ϕ_i is the angle of elevation of the line segment between point p_i and p_{i-1} and $\chi_i = \phi_i - \phi_{i+1}$ for $0 < \chi < \pi$.

Elastic matching associates a point in the model with each point of the unknown. This point pairing occurs when we evaluate $d(i,j;k)$ associating a given unknown point i with a specific model point j. The algorithm is allowed to look at a finite number of model points, and keeps the one closest according to the distance measurement $d(i,j;k)$. With similar patterns, unknown points will be associated with model points that are relatively close to the order in which they appear e.g.: the 10th unknown point with the 12th model point. Essentially, the algorithm will associate together points lying on the diagonal between $(0,0)$ and (n,m) for an unknown of n points with a model of m points. The diagonal is then the idealistic pairing one would hope to achieve. To visualize this we can plot a surface where the elevation of the surface point represents the distance between a point of an unknown pattern and a given point of a model. In Figure 3 we notice that the points near the diagonal usually have the lowest distance. The elasticity makes it possible to handle small variations in shape as well as to compare patterns containing different numbers of points.

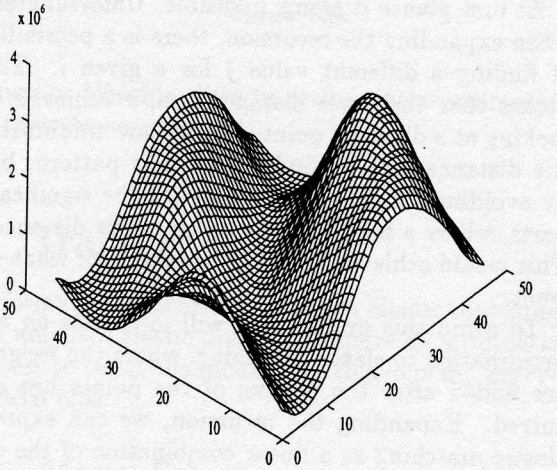

Figure 3: Distance surfaces of two '0' (zero) patterns

Elastic matching removes the cost typically associated with the substitution operation used in the string matching algorithm. Instead, elastic matching imposes a constraint as to how far down the string one can search. When the algorithm is allowed to use a deeper look-ahead, point pairing strays farther from the diagonal, minimizing the distance by comparing with fewer model points that are farther and farther apart. This, in the end, creates more confusion than anything else and harms the recognition performance. Although restricting the look-ahead to a depth of two appears arbitrary, experimentation with other values showed that a look-ahead of two was optimal [6].

3.2 Reformulation of Elastic Matching

With elastic matching, each matched point contributes an equivalent amount to the total distance. It is frequently desirable in character recognition algorithms to give some portions of a pattern more importance than others. To do so we want to add weights to certain points of the models to increase or decrease their contribution to the total sum, giving the corresponding section more or less importance. We may then be tempted to add the weights to each of the distance terms $d(i,j;k)$ in Equation 7 as follows:

$$D(i,j;k) = \omega_j d(i,j;k) + min \left\{ \begin{array}{l} D(i-1,j;k) \\ D(i-1,j-1;k) \\ D(i-1,j-2;k) \end{array} \right\} \quad (9)$$

At first glance it seems plausible. Unfortunately, when expanding the recursion, there is a possibility of finding a different value j for a given i. This means that the lower distance is now achieved by looking at a different point. We are now minimizing the distance, not by finding the best pattern, but by avoiding, in part, matching with the significant parts, where a high weight creates larger distances. This would achieve exactly the opposite of what we want.

To avoid this problem we will formulate an approximation to elastic matching, where the weights are added after the pairing of the points has occurred. Expanding the recursion, we can express elastic matching as a linear combination of the coefficients:

$$D(i,j;k) = d(0,a_0;k) + d(2,a_2;k) + \ldots + d(i,a_i,k) \quad (10)$$

where a_i is a point in the model k and on which the following restriction is imposed:

$$a_i = j \quad (11)$$

This expansion creates the following relationship between the model and the unknown pattern that associates with each point of the unknown pattern a point a_i in the model. The value of a_i is given by the elastic matching algorithm. The evaluation of the of the elastic matching, described in Equation (7), results in a pairing of points between the ith unknown point and the j point of the model, which we refer to as a_i.

We can now express elastic matching as a linear combination of distance measurements:

$$D(i,j;k) = \sum_{i=1}^{n} d(i,a_i;k) \quad (12)$$

3.3 Weighted Elastic Matching

With this representation of elastic matching the pairing of points is a process independent of the weights associated with a specific point. This formula now behaves as we would like, the weights not interfering with the tracking properties of the algorithm. The higher the weight at a given point, the more critical a close match to the model is at that point.

$$D(i,j;k) = \sum_{i=1}^{n} \omega_{a(i)} d(i,a(i);k) \quad (13)$$

Two things need to be mentioned about this representation of elastic matching. First, the formulation now bears some resemblance to simple neural networks such as the perceptron, linking this algorithm with a well studied class of algorithms and pointing at an avenue to improve it. Secondly, the implementation is not significantly different from traditional elastic matching. It can be shown that the implementation for Equation (7) yields the same results as Equation (12) for similar patterns. There may be some differences between the two equations for patterns that are very different, but this is of little consequence in our case. When two patterns are so different that they yield different distance measurements with our formulation, their distance will be much higher than the recognition threshold. As a result, pairs of patterns with high distances will be rejected as candidates for recognition irrespective of the distance measurement formula used. The recognition experiments performed with both versions of the algorithm confirm this observation.

The computational cost of adding weights is relatively small. This can be understood by the fact that the weight is added after we determine which model point matches with an unknown point. Most of the computational cost was incurred in calculating the distance $d(i,j;k)$ between one point i and three potential candidates j. This operation involves, expressed by Equation (8), several additions and multiplications. The resulting additional cost does not represent a serious performance cost.

3.4 Weight Calculation

We need to create the weights. Basically we need a function that will give a high weight when a given model point is a good discriminator between the

two classes and a low weight otherwise. To do so we need to create, for each potential model, a set of unknowns of the correct set and of the error set.

These sets are selected from the patterns that are close to the model. The patterns kept are those falling below the recognition threshold for the correct set and above three times the recognition threshold for the error set. Once we have obtained the needed sets, we evaluate the average distance, from points of the unknowns to the points of the models as follows:

$$\overline{dtot}(j;k,set) = \sum_{\forall l \in set} \omega_j dtot(j;k,l) \qquad (14)$$

as $dtot(j;k,l)$ is:

$$dtot(j;k,l) = \sum_{i=0}^{n} \left\{ \begin{array}{ll} 0 & a_i \neq j \\ d(i,j;k,l) & a_i = j \end{array} \right\} \qquad (15)$$

and is the sum of all of the points in an unknown patterns matched with a given point j in the model l. The average distance $\overline{dtot}(j;k,set)$ for both the error and correct set gives the results shown in Figure 4.

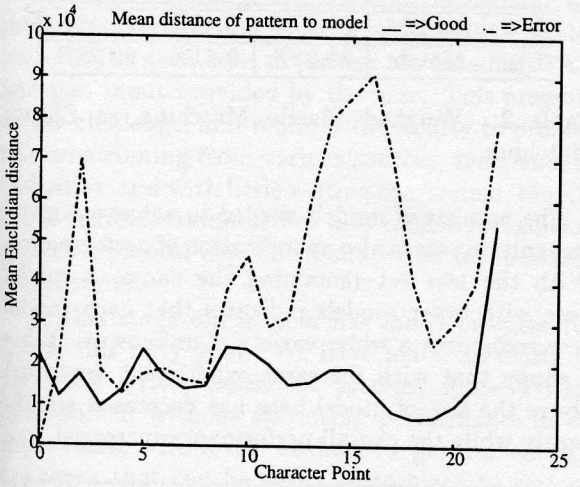

Figure 4: Average distance for each point from both sets to a model.

To measure how good a discriminator a point is, we use the ratio of the average distance of the error points over the average distance of the correct points. Equation (16) behaves as needed, as it grows when the distance between the two sets increases.

$$\omega_j = \overline{dtot}(j;k,err)/\overline{dtot}(j;k,correct) \qquad (16)$$

This gives weights that behave as needed: weights are high when the distance between two classes is high and low when the correct class and error classes are close as illustrated by Figure 5.

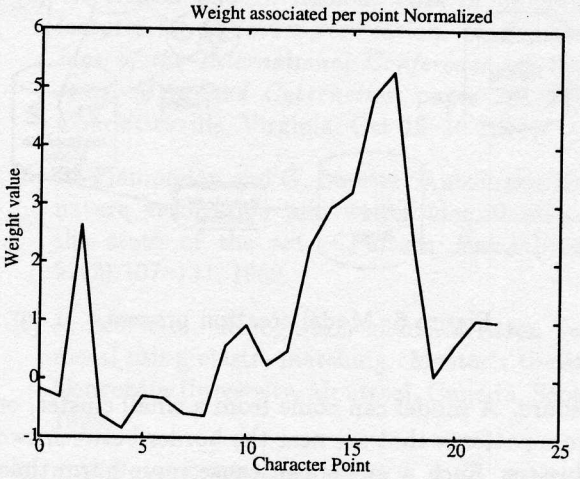

Figure 5: Weights given to each point as estimated by *error/correct* set.

4 Training Process

Performance comparisons between elastic matching and weighted elastic matching require an automatic process to select the model base to avoid any experimental bias.

4.1 Model Selection

The models are patterns taken directly from training sets. There are two main steps involved in model selection as illustrated in Figure 6: a *clustering* step and a refining step called a *bad model removal.*

The first step is to create clusters containing patterns that give us potential models for a given class. The clusters are fully connected, indicating that each member is below the recognition threshold to all the other members of the cluster. Each member of the cluster can represent the entire cluster but we use the centroid as being the most representative of the cluster. We define the centroid as the element with the lowest average distance to all the other elements of the cluster as expressed by Equation (17) for a cluster containing n distinct elements.

$$D_{centroid} = \min_{k=1..n} \sum_{i=1}^{n} D_{k,i} \qquad (17)$$

But in the second step these potential models must be weeded out with a bad model removal pro-

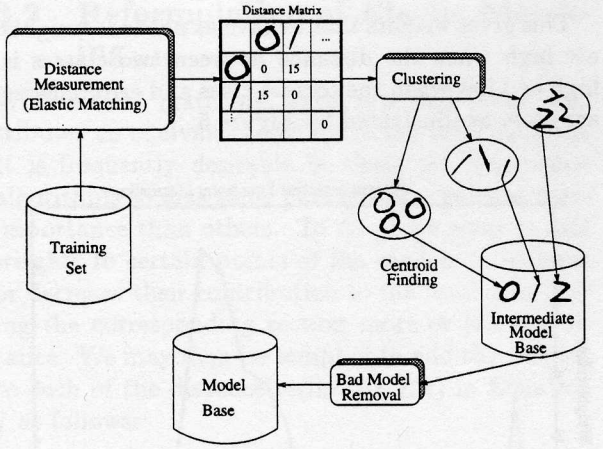

Figure 6: Model creation process.

cedure. A model can come from a small cluster, or from patterns that are near the border between two clusters. Such a model can cause more harm than good to the recognition process and should be removed. A model is judged to be a bad model if the following is true:

$$R - (S + C) < 0 \qquad (18)$$

4.2 Iterative Training

Once we have calculated the weight, we hope that the distance measurement will change for the better. This implies that some patterns will move in closer, while others will move farther away. This means that both the correct set and the error set used to calculate the weights will change if we attempt to calculate the weights again. Hence the weights are no longer optimal. To remedy this the calculation of the weight is performed over many iterations. This iterative weight calculation process is illustrated in Figure 7.

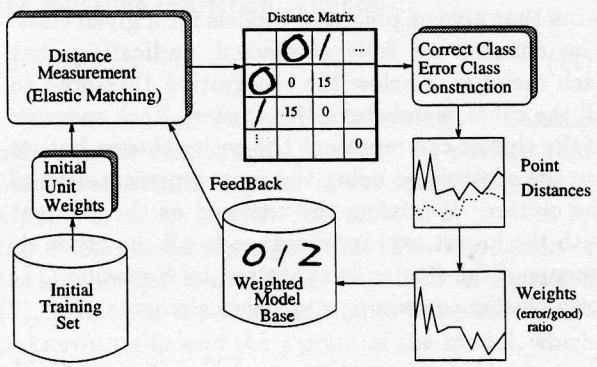

Figure 7: Iterative process used to refine weights

5 Recognition Results

Various experiments were performed with the online database described above. We performed the same recognition experiment with both Elastic Matching and Weighted Elastic Matching, in an attempt to determine if weighted elastic matching improves performance, primarily by looking for an increase in the recognition rate.

5.1 Experiments Performed

Experiments were performed with the various data sets. We keep the iteration that displays the best recognition results with the training set and use the model base generated for that step for recognition with the testing set.

5.2 Experimental Results

Table 2 compares the recognition result of the weighted elastic matching against the regular formulation of elastic matching. As we can see, weighted elastic matching improved the recognition rate while maintaining reliability.

Method	Reco	Subs	Relia
Weighted EM	88.67%	0.48%	99.46%
Elast. Match.	85.22%	0.61%	99.29%

Table 2: Weighed Elastic Matching *vs* Elastic Matching.

The number of models needed to achieve a given recognition rate is also an indication of performance. With the test set remaining the same, a model base with fewer models indicates that each model now recognizes a wider variety of unknowns. Table 3 shows that with the same experiment described above the size of model base has decreased significantly while the overall performance increased.

	Weighed EM	Elast. Match.
Nb. Models	293	337

Table 3: Model base size.

The size of the model base is still large considering the number of patterns recognized. However other experiments showed that the majority of recognition is performed by a relatively small subset of the model base.

Various other experiments were performed with scanned data. In this case more preprocessing has to be performed to convert the scanned imaged into

a sequence of points. The obtained point sequence is treated just like the tablet data. Current results are reasonable and indicate that it is possible to apply string matching methods to tablet data.

No experiments were performed with an alphanumeric database. The original elastic matching algorithm was used with the full alphabet, so we expect little difficulties in transfering over these results to the weighted elastic matching. This, however, is not to discount the new difficulties usually encountered with the alphabet. Some pairs of symbols are not reliably distinguishable without the context, e.g.: '0'(zero) and 'O'(oh), '1' (one) and 'l' (ell). Some forms of script writing use the same shapes for the lower and upper case letters either by varying their size or modifying their position relative to the baseline. Most recognition systems typically make few assumptions about the position or size of a character and have difficulties in distinguishing such cases.

6 Conclusion

More and more software is built from a multiple set of tools and libraries, each offering specific functionality. The ongoing integration of information processing technology will result in the demand for integrated recognition of both scanned and tablet data, as the need arise to process the incoming FAX and pen input provided by the user. This presents a new challenge, and requires the ability to process patterns coming from various sources, independent of format, and with little writing constraints, requiring a flexible, trainable but also robust recognition system for writer-dependent and writer independent applications.

At this stage our system has made progress towards this lofty goal. We have made an existing, primarily writer-dependent system, more writer-independent without sacrificing performance in either arena. However we have a general enough approach that can be easily customized to various performance requirements or to different sources of data.

References

[1] J.B. Krushkal. An overview of sequence comparison: Time warps, string edits, and macromolecules. *SIAM review*, 25(2):201–237, 1983.

[2] R. Legault and C.Y. Suen. Smoothing of 2-d binary contours. Technical report, June 1993.

[3] Steven Malowany. A graphical rule-based system for recognizing on-line digits. Master's thesis, Concordia University, Montreal, Canada, April 1993.

[4] W. Nelson and E. Kishon. Use of dynamic features for signature verification. In *Proceedings of the International Conference on Systems, Man, and Cybernetics*, pages 201–205, Charlottesville, Virginia, Oct 13–16 1991.

[5] R. Plamondon and G. Lorette. Automatic signature verification and writer identification–the state of the art. *Pattern Recognition*, 22(2):107–131, 1989.

[6] P. Scattolin. Recognition of handwritten numeral using elastic matching. Master's thesis, Concordia University, Montreal, Canada, Sept. 1993.

[7] C.Y. Suen, S. Malowany, and S. Adel. Pattern recognition and expert system. In *Proceedings, Vision Interface '93*, pages 149–156, Toronto Ontario, May 18–21 1993.

[8] C.Y. Suen, C. Nadal, R. Legault, Mai T.A., and L Lam. Computer recognition of unconstrained handwritten numerals. *Proceedings of IEEE*, 80(4):1162–1171, July 1992.

[9] C.C. Tappert. Cursive script recognition by elastic matching. *IBM Journal of Research and Development*, 26(6):765–771, November 1982.

[10] C.C. Tappert. Speed, accuracy, flexibility trade-offs in on-line character recognition. Research Report RC13228, IBM, October 1987.

[11] C.C. Tappert, C.Y. Suen, and T. Wakahara. The state of the art in on-line handwriting recognition. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 12(8):787–807, 1990.

[12] J.R. Ward. One view of on-going problems in handwriting character recognition. In *Frontiers in Handwriting Recognition*, pages 181–194, CENPARMI Concordia University, Montreal, Canada, April 2-3 1990.

[13] J.R. Ward and M. Philips. Digitizer technology: performance characteristics and the effects on the user interface. *IEEE Computer Graphics and Applications*, 7(4):31–44, April 1987.