An Additive On-line Single Character Recognition Method

Jakob Sternby

An Additive On-line Single Character Recognition Method

Jakob Sternby
jakob@maths.lth.se
Centre for Mathematical Sciences
Sölvegatan 18, Box 118
S-221 00, Lund, Sweden

Abstract

The on-line isolated character recognition problem is in many respects a mature research area with several industrial applications. When it comes to on-line cursive word recognition progress has been much slower. One of the obstacles refraining simple extension of the validated single character recognition methods to cursive word recognition is the fact that most methods rely on various forms of global normalization techniques which makes it difficult to define a fair *way of adding such matching distances for sequences. This paper proposes a new single character recognition method where focus lies on ensuring the additive property such that it can easily be extended to be used for on-line cursive character recognition in a well-defined way. Various experiments with the new methodology have been conducted on the UNIPEN database with promising results.*

1 Introduction

The problem of recognizing on-line isolated single characters is a mature research area which has also been proven in industrial applications. An extensive survey and an overview of different approaches to the problem can be found in [8]. In recent years Neural Networks and statistical models based on Hidden Markov Models have been popular strategies for character recognition but they do not necessarily provide a higher recognition accuracy than DP-based methods [1, 2, 6].

Extending the task to recognition of cursive words introduces a difficult layer of complexity in the form of segmentation. Some of the best results so far have been attained by methods employing various kinds of Time-Delay Neural Networks [4, 9]. A critical issue when constructing a cursive recognizer from a method originally aimed for the single character problem is the problem of how to add the single character recognition scores from various segmentation paths so that the different word hypothesis are treated fairly. Since most single character recognition methods involve some kind of normalization of the input and a set of shorter models would generally be favored since the constituent parts would get a better alignment. Many methods have treated this problem in ad hoc manners by providing a *summing procedure* that provided *good* results [7], or by using extremely simplistic additive matching methods [11].

This paper presents a new additive single character recognition method. The method is based on the principles of Frame Deformation Energy as presented in [10], where each *stroke* (the sequence of coordinate between a pen-down and pen-up event) is subdivided into segments according to a simple segmentation strategy based on the direction orthogonal to the writing direction. The additivity property of the novel method is maintained by the fact that each segment is compared to the database by normalization strategies that are independent of the segmentation. It will be shown that with this method a concatenation of database models matched to a longer input segment (i.e. a cursive word) differs from the matching of the same models in a connected state to the same segment by a simple operation. Thus the novel method possesses the additivity property necessary to expand the method to on-line cursive handwriting recognition. The theory has been validated on the UNIPEN database with very promising results.

2 Additive distance functions

The recognition methods developed for *single character recognition* (SCR) that exist today have been optimized to maximize recognition accuracy in this idealized recognition setting. The extension of such recognition methods to counter the problem of on-line cursive handwriting recognition has however proven to be a very difficult task. This paper proposes a different strategy aiming at ensuring the additive property, as defined below, of distance functions to facilitate the development of more powerful recognition methods for cursive on-line handwriting recognition.

In segmentation based cursive handwriting recognition a single character recognition method is used to find the best interpretation of a segment of the cursive word. Here the segmentation of a handwritten set of strokes $X = \{x_j\}$ will be denoted by $\mathcal{S}(X) = \{\mathcal{S}(x_j)\} = \{\{s_i^{x_j}\}_{i=1}^{|\mathcal{S}(x_j)|}\}_{j=1}^{n}$, where $s_1^{x_j}$ is the first of the $|\mathcal{S}(x_j)|$ segments of stroke j. Segmentations of two different sets of equal number of n strokes (X, Y) are said to be *similar*, $\mathcal{S}(X) \sim \mathcal{S}(Y)$ if $|\mathcal{S}(x_j)| = |\mathcal{S}(y_j)|, j = 1, \ldots, n$. The space of all labeled sets of sampled strokes of on-line handwriting is denoted by $\mathbb{X}$.

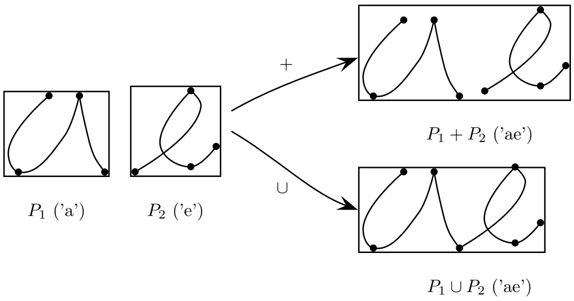

Figure 1: The difference between addition and concatenation of two samples P_1 and P_2 with the resulting samples and their class labels to the right.

To clarify the process of matching several prototypes against one set of strokes two operations in sample space $\mathbb{X}$ are introduced. The additive operator $+$ is used for adding two separate sample segments P, Q by letting the first point of the second sample constitute the starting point of a new stroke. The concatenation operator $\cup$ will denote the connection of two sample segments P, Q by attachment of the first point of the second segment to the last point of the first. The difference between these operations are illustrated in Figure 1. The placement of the second segment resulting from the $+$ operation is a parameter that can be specified by the implementor.

Definition 1 *A distance function* $d(\cdot,\cdot) : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$ *is* additive *if for any four samples* $P, Q, X_P, X_Q \in \mathbb{X}$ *such that* $\mathcal{S}(P) \sim \mathcal{S}(X_P)$ *and* $\mathcal{S}(Q) \sim \mathcal{S}(X_Q)$ *there exist uniquely defined functions* $\alpha(P, Q, X_P, X_Q)$ *and* $\beta(P, Q, X_P, X_Q)$ *such that*

$$\begin{aligned} d(P, X_P) + d(Q, X_Q) + \alpha(P, Q, X_P, X_Q) \\ = d(P + Q, X_P + X_Q) \end{aligned} \tag{1}$$

$$\begin{aligned} d(P, X_P) + d(Q, X_Q) + \beta(P, Q, X_P, X_Q) \\ = d(P \cup Q, X_P \cup X_Q). \end{aligned} \tag{2}$$

3 Additive FDE

The segmentation strategy in this paper is identical to that of [10]. This segmentation strategy assumes that the writing direction is known (or calculated with some reliable method). Segmentation points are then placed on the curve as the local extreme points that are *deeper* than some fixed threshold. This crude strategy is actually all that is needed to produce the segmentation $\mathcal{S}(X)$ referred to in Section 2 and also called the *frame* of the sample. Each such segment is then parameterized by the *Djikstra Curve Maximization* (DCM) strategy with 3 intermittent points. This technique chooses the chosen points on the curve in such a way that the resulting curve length is maximized.

3.1 Segmental features

In this paper the original ideas of the Frame Deformation Energy (FDE) concept of segmental character recognition have been adjusted to construct an additive distance function. A novel curve distance function has also been developed to improve recognition accuracy. The FDE single character recognition method divides an input character into segments and applies both local and segmental feature extraction as input to the recognition process [10]. In this paper these features have been somewhat reworked and some new features have been introduced. For each segment s in a segmentation $\mathcal{S}(X)$ of a sample X the *normalized direction* $a(s) \in S^1$ is calculated. Each subsequent pair of connected segments s_i, s_{i+1} also generate the features

- *Normalized length ratio* $\lambda(s_i, s_{i+1}) \in (0, 2)$
- *Normalized connection angle* $\phi(s_i, s_{i+1}) \in S^1$

Subsequent segments that are non-connected i.e. between strokes also use the *Normalized length ratio* $\lambda(s_i, s_{i+1})$ but also generate an additional separate set of features pertaining to relative positioning of the next stroke.

- *Normalized vertical position* $\kappa(s_i, s_{i+1}) \in [0, 1]$
- *Normalized segment gap* $\delta(s_i, s_{i+1}) \in (-1, 1)$

The different features are illustrated along with their normalization procedures in Figure 2.

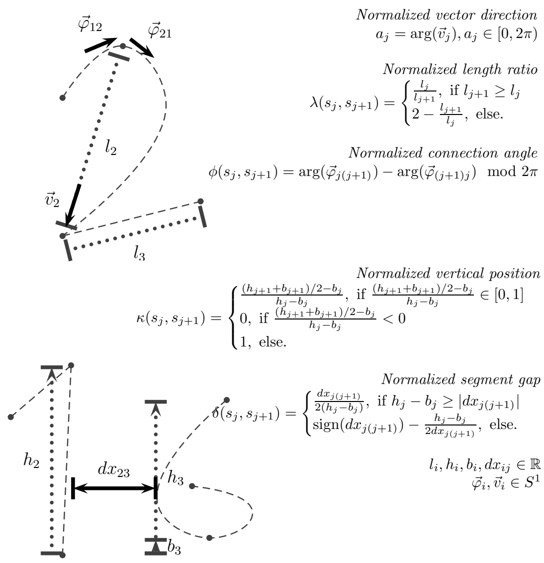

Figure 2: A graphic example of the segmental features described in Section 3.1.

3.2 Distance function

The distance function presented in this paper is based on the features of Section 3.1 in combination with a segmental curve distance function acting on the transformed segments resulting from translation, angle and length transformation. This paper presents a new curve distance function DCM-DTW based on dynamic programming constructed to deal with the specific issues of discriminating between handwritten curves. Since translation, angle and length differences between the segments have been removed this function measures the distance between two curve segments aligned so that they share start and endpoints.

3.2.1 DCM-DTW

The curve distance function presented here is called DCM-DTW since it is a Dynamic Time Warping (DTW) influenced distance function developed to discriminate well between curves parameterized according to the DCM method presented in Section 3. Points placed with DCM are spaced unevenly on the curve as this method focuses on retaining the shape information and not on providing a smooth parameterization. For this reason a dynamic programming method matching two such point configurations need to allow Point-to-curve matching in addition to traditional Point-to-point matching. Furthermore as the number of points are few compared to traditional arclength parameterizations, the directional vector used in many implementations [1, 6], is no longer a stable feature. Instead DCM-DTW makes use of the intermittent angle θ_p of a point p defined as $\theta_p = \arg(p_{\epsilon+} - p) - \arg(p - p_{\epsilon-}), \theta_p \in [0, 2\pi)$, where $p_{\epsilon+}, p_{\epsilon-}$ denote the next and the previous points at a distance ϵ on the curve.

To accomplish the desired flexible Point-to-Curve matching, the closest point on each line segment of the opposing curve is calculated for each point. Let $L_j(t) = tp_{j-1} + (1-t)p_j, t \in [0,1]$ denote the line segment between points p_{j-1}, p_j on curve $P = \{p_k\}$. Let $t_{q_i}^{L_j} = \operatorname{argmin}_{t\in[0,1]} \|L_j(t) - q_i\|$ for a line segment $L_j(t)$ on P and a point q_i on Q. With this notation the *pseudo* points $x_{j,r}$ on the curve P w.r.t. the curve Q, both with n points are defined as $x_{j,r}^{P,Q} = L_j(t_{q_r}^{L_j})$.

A basic distance function between points or pseudo points (x, y) is introduced as $\mathfrak{g}(x,y) = \|x-y\| + k_\theta\|\theta_x - \theta_y\|$, where k_θ is a normalization constant for balancing angle distance with coordinate distance.

A dynamic programming distance function based on this premise has an alignment function $\Phi(k) = (\phi_p(k), \phi_q(k))$ and transitions $(1,0), (1,\xi), (\xi,1), (0,1), (1,1)$ where the novel $(1,\xi)$ and $(\xi,1)$ transitions mark the transition to or from the *pseudo* points in P or Q respectively. The alignment function has the requirements that $\Phi(1) = (1,1), \Phi(m) = (n,n)$. When adding transitions to the alignment functions the addition of two ξ is defined by $\xi + \xi = 1$. The alignment state $(\phi_p(r), \phi_q(r)) = (k+\xi, j)$ is defined as $(p_{\phi_p(r)}, q_{\phi_q(r)}) = (p_{k+\xi}, q_j) = (x_{k+1,j}, q_j)$. The distance function $d_{DCM}(\cdot,\cdot)$ can now be formulated as

$$d_{DCM}(P,Q) = \min_{\Phi} \sum_{i=1}^{m} \mathfrak{g}(p_{\phi_p(i)}, q_{\phi_q(i)}) \tag{3}$$

where the transitions $(1,\xi), (1,1), (1,0)$ are disallowed from states $(\phi_p(r), \phi_q(r)) = (k+\xi, j), k, j = 1, \ldots, n$ and similarly $(\xi,1), (1,1), (0,1)$ are disallowed from states $(\phi_p(r), \phi_q(r)) = (k, j+\xi), k, j = 1, \ldots, n$.

As pointed out in [6] conventional DP-algorithms for matching handwritten characters suffer from over-fitting the prototype to the sample, and to improve the situation the simple distance function $\mathfrak{g}(\cdot,\cdot)$ has been updated with a weight function $f(\cdot) : \mathbb{R} \to \mathbb{R}$ which also considers the situation in which the points differ. The over-fitting problem for conventional DTW arises from the fact that samples of classes with very curved strokes such as the digit '3' differ much more than classes with straight strokes such as the digit '1'. Here the special setting in which d_{DCM} is applied (e.g. the curves share start and endpoints). In this paper the weight function $f(x) = 0.2x^2 - 1.1x + 1, x \in [0,1]$ has been tested by applying it as according to the minimal Euclidean distance of the point pair (x, y) to their common baseline defined by the start and endpoints. Let the baseline be the line $\mathbf{b}(t) = p_1 + \vec{v}t, \vec{v} = p_n - p_1$ and define $\nu_{x,y} = \operatorname{argmin}_{v\in\{x,y\}} d_L(\mathbf{b}, v)$, where d_L is the orthogonal distance between the point v and the line $\mathbf{b}$. The modified distance function $\mathfrak{g}_{DCM}$ then becomes

$$\begin{aligned}\mathfrak{g}_{DCM}(x,y) = f(\min(\nu_{x,y}, \|p_n - p_1\|))\cdot \\ (\|x-y\| + k_\theta\|\theta_x - \theta_y\|).\end{aligned} \tag{4}$$

The complete algorithm between two segments $P = \{p_j\}_{j=1}^n$ and $Q = \{q_j\}_{j=1}^n$ can now be formulated as in Algorithm 1.

Algorithm 1 DCM-DTW

for $i, j := 1, \ldots, n$ **do**
  if $i < n$ **then**

$$d(i_\xi, j) := \mathfrak{g}(x_{i,j}^{P,Q}, q_j) + \min \begin{cases} d(i, j-1) \\ d(i_\xi, j-1) \\ d(i, (j-1)_\xi) \end{cases}$$

  end if
  if $j < n$ **then**

$$d(i, j_\xi) := \mathfrak{g}(p_i, x_{j,i}^{Q,P}) + \min \begin{cases} d(i-1, j) \\ d(i-1, j_\xi) \\ d((i-1)_\xi, j) \end{cases}$$

  end if

$$d(i,j) := \min \begin{cases} d(i-1, j) + \mathfrak{g}(p_i, q_j) \\ d(i-1, (j-1)_\xi) + 2\mathfrak{g}(p_i, q_j) \\ d(i, (j-1)_\xi) + \mathfrak{g}(p_i, q_j) \\ d(i, j-1) + \mathfrak{g}(p_i, q_j) \\ d((i-1)_\xi, j-1) + 2\mathfrak{g}(p_i, q_j) \\ d((i-1)_\xi, j) + \mathfrak{g}(p_i, q_j) \\ d(i-1, j-1) + 2\mathfrak{g}(p_i, q_j) \end{cases}$$

end for
$d_{DCM}(P,Q) := d(n,n)/2n$

3.2.2 Additivity and FDE

Unlike most single character recognition methods the FDE has an intrinsic additive property in that it treats segments in an additive way by utilizing a linear distance function of feature components that are normalized independently in sample and database. The components can be subdivided into segmental d_S and connective d_C components, where the segmental distance components compares two segments and the connective components compares the connection between two pairs of segments. In this paper the segmental distance component between two segments (s_1, s_2) has been defined as

$$d_S(s_1, s_2) = w_A|a(s_1) - a(s_2)| + w_P d_{DCM}(A_{12}(s_1), A_{21}(s_2)), \tag{5}$$

where one of the operators A_{12}, A_{21} is the identity operator and the other operator aligns the start and endpoint of the smaller segment to the longer segment. Figure 3 shows an example of where all segments of a database prototype have been aligned by such transformations to the sample.

The distance component d_A between two pairs of attached segments $(s_{11}, s_{12}), (s_{21}, s_{22})$ is similarly defined as

$$d_A((s_{11}, s_{12}), (s_{21}, s_{22})) = w_L|\lambda(s_{11}, s_{12}) - \lambda(s_{21}, s_{22})| + w_C|\phi(s_{11}, s_{12}) - \phi(s_{21}, s_{22})|. \tag{6}$$

Finally there is a distance component for non-connected segments derived from the *Normalized length ratio* as well as the non-connected features *Normalized vertical position* and the *Normalized segment gap* as specified in Section 3.1. This distance component is denoted by d_N and is defined as

$$d_N((s_{11}, s_{12}), (s_{21}, s_{22})) = w_L|\lambda(s_{11}, s_{12}) - \lambda(s_{21}, s_{22})| + w_V|\kappa(s_{11}, s_{12}) - \kappa(s_{21}, s_{22})| + w_G|\delta(s_{11}, s_{12}) - \delta(s_{21}, s_{22})|. \tag{7}$$

The distance component $d_C((s_{11}, s_{12}), (s_{21}, s_{22}))$ for a connection between two segments can now be defined by combining Eq. 6 and Eq. 7 as

$$d_C(\cdot, \cdot) = \begin{cases} d_A(\cdot, \cdot) \text{ if segments are connected,} \\ d_N(\cdot, \cdot) \text{ else.} \end{cases} \tag{8}$$

The total additive FDE distance function $d_{FDE}(X, Y)$ between two samples with segmentations $\mathcal{S}(X) = \{s_i^X\}_{i=1}^{|\mathcal{S}(X)|}$ and $\mathcal{S}(Y) = \{s_j^Y\}_{j=1}^{|\mathcal{S}(Y)|}$ such that $\mathcal{S}(X) \sim \mathcal{S}(Y)$ can now be stated as

$$d_{FDE}(X, Y) = \sum_{i=1}^{|\mathcal{S}(X)|} d_S(s_i^X, s_i^Y) + \sum_{i=1}^{|\mathcal{S}(X)|-1} d_C((s_i^X, s_{i+1}^X), (s_i^Y, s_{i+1}^Y)). \tag{9}$$

Obviously this distance function is dependent on the weights $(w_A, w_P, w_L, w_V, w_G)$ in Eqs. (5), (6) and (7). For the additivity discussion, however, it will temporarily be assumed that these weights have been set to some predefined value. A small discussion on the optimization of these weights follows in Section 3.2.4.

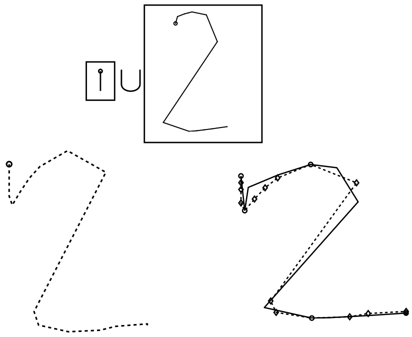

Figure 3: A plot of one of the samples of class '2'(dashed) that found a best match to a concatenation of a simple noise prototype with a prototype for '2' in the additive experiment of Section 4.4. The lower right shows the aligned segments of the concatenated model to the sample.

3.2.3 Additivity

It follows almost immediately from the components of Eq. 9 that the novel distance function is additive according to Definition 1. Consider a sample X of points with n segments $\{s_j^X\}_{j=1}^n$ distributed over m strokes and two prototypes (samples) P, Q such that $\mathcal{S}(P) \sim \{s_j^X\}_{j=1}^{j_{PQ}}$ and $\mathcal{S}(Q) \sim \{s_j^X\}_{j=j_{PQ}+1}^n$ for some segment index j_{PQ}. For simplicity let $X_P = \{s_j^X\}_{j=1}^{j_{PQ}}$ and $X_Q = \{s_j^X\}_{j=j_{PQ}+1}^n$. If these segments are matched individually to each other the sum $d_{FDE}(P, X_P) + d_{FDE}(Q, X_Q)$ can be written as

$$d_{FDE}(P, X_P) + d_{FDE}(Q, X_Q) = \sum_{i=1}^{j_{PQ}} d_S(s_i^P, s_i^X) + \sum_{i=j_{PQ}+1}^{n} d_S(s_{i-j_{PQ}}^Q, s_i^X) \sum_{i=2}^{j_{PQ}} d_C((s_{i-1}^P, s_i^P), (s_{i-1}^X, s_i^X)) + \sum_{i=j_{PQ}+1}^{n-1} d_C((s_{i-j_{PQ}}^Q, s_{1+i-j_{PQ}}^Q), (s_i^X, s_{i+1}^X)) \tag{10}$$

It is easy to see that the right hand side of (10) is very close to being the definition of the distance function for a combined model of P, Q to the whole sample X. In fact the only differing part is clearly the distance between the actual connecting segments $d_C((s_{j_{PQ}}^P, s_1^Q), (s_{j_{PQ}}^X, s_{j_{PQ}+1}^X))$. If these segments are connected then clearly setting $\beta(P, Q, X_P, X_Q) = d_A((s_{j_{PQ}}^P, \mathrm{T}_P(s_1^Q)), (s_{j_{PQ}}^X, s_{j_{PQ}+1}^X))$ assures property specified by (2), where T_P is the translation operator to the last point of P. Similarly the property specified by (2) is assured if the equivalent translation operator T_{P+} is the same as the one used for placing Q when adding two samples $P + Q$ by the $+$ operation.

3.2.4 Component Weights

A problem that arises when differentiating a distance function into separate components is the issue of how to balance the output of every component. This paper does not focus on the optimal setting of the weights that appear in Eqs. (5), (6) and (7) however a coarse and workable estimation may be obtained by viewing the weights as a hyperplane and determining this hyperplane by SVM. One way of doing this which was done to obtain an initial value for all weights is to produce one *positive* element and one *negative* set of distance components for each sample in the training set. The *positive* element was obtained as the set of distance components between a sample and the cluster center which the sample belonged to and correspondingly the *negative* element was obtained as the distance components between the sample and the cluster center of a neighboring class. The initial estimation of the weights was then obtained as the hyperplane obtained through *LinearSVC* as implemented in the `osu-svm` package [5].

A further merit of using linear distance components with weights is that it is very easy to define secondary *zoom* functions to further differentiate in recognition between top candidates in the recognition output, this however has not been experimented with in this paper.

4 Experiments

The experimental part of this paper can be divided into three separate experiments. The first part evaluates the novel segmental distance function d_{DCM} described in Section 3.2.1. The second part is focused on the performance of the FDE described here in a conventional single character recognition context. Finally an experiment evaluating the additive property has been conducted in a single character context.

4.1 Data set

The data set used in all experiments is the 1a part of the UNIPEN database containing on-line handwritten digits [3] since it has been a popular choice in recent publications [1, 2, 6]. The data set was divided into one third test set and two-thirds training set using the same tools as described in [6]. However, unlike the previous papers preprocessing included removal of some samples (observed by clustering) of the classes '0', '4' and '8' which were mislabeled (due to erroneous stroke segmentation). The motivation for this is that this paper does not present a method that primarily is involved with automatic training where such samples might be present. However, no samples were removed from the test set since any method needs to deal with bad and good input in the recognition phase. The preprocessing also included segmentation according to Section 3. Some extremely short segments bound to be noise were also removed at this stage.

4.2 Evaluation of DCM-DTW

It is of course difficult to evaluate the performance of the segmental curve matching method individually. Since it should discriminate well between single segments all samples consisting of single segments were extracted both from the training set and the test set. A recognition test was made using all allographs in the training set to construct a 1-nn classifier. The novel DCM-DTW method was compared to the conventional DTW presented in [1]. The methods were compared both using the DCM-style normalization of aligned endpoints and with the mass-center normalization customary for other DP-matching methods. The experiments revealed that DCM-DTW performed better in the DCM-normalization case but also that the additive FDE on single segments actually performed well even compared to the normal DTW based on 32 evenly spaced and mass center aligned points (results in parenthesis in Table 1). This shows that the DCM-DTW is a good choice for the segmental curve matching and that it is probable that the DCM-parameterization techniques contains sufficient shape information on each segment.

Table 1: Recognition results on the single segment samples of the UNIPEN/1a data set (includes 469 single segment test digits of '1','2','3','5', '7' and '9')

Method	Top-1	Top-2
DCM-DTW	**98.93%**	100%
DTW (MC)	97.87% (99.57%)	99.79% (99.57%)

4.3 Conventional SCR

The intrinsic composition of the additive FDE distance function makes it sensitive to noise in the form of extra segments. The segmental features of such extra noisy segments is highly stochastic and is thus likely to cause large distance variations of some samples of the same class. Similar samples of different classes therefore also run the risk of being erroneously determined by the matching of the noisy parts instead of the parts belonging to the actual character shapes. This property suggests that noise should be treated in another way than including it in the database models. The next section shows a simple but effective remedy to the problem. A next step in this research is therefore to develop methods for removing the noise segments from the training set. For completeness, however, some automatic tests have been carried out and the results are presented below. The tests were conducted by using the clustering algorithm presented in (reference omitted for anonymity reasons) which produces a clustering well suited for recognition. Since the method is sensitive to extra noise segments the method requires a large number of clusters to approach the state-of-the-art DP-matching methods accounted for in [1, 6]. On the other hand the definition of the similarity for segmentations limits the maximal number of actually compared prototypes as seen in Table 2. Furthermore the FDE-method presented here requires less storage and is faster and for that reason a larger number of prototypes may be acceptable.

Table 2: Recognition results by using the Additive FDE as a conventional SCR

Prototypes	Max Compared	Top-1	Top-2
228	33	85.74%	93.33%
491	135	91.46%	96.58%

4.4 Additive SCR with noise model

In this experiment two additional simplistic noise models consisting of a single straight segment were added. The aim of this experiment was both to validate the additive principles of the recognition method presented in this paper as well as to show that the novel method implemented in this way has potential also for top single character recognition results. The recognition was implemented using the recognition graph technique presented in [11] where the dictionary only consisted of single digits. Since the noise models were not given a class the validity checks assured that any final recognition results corresponded to exactly one digit prototype combined with zero or more noise prototypes (via the $+, \cup$ operations explained in Section 2). The recognition test was conducted on the initial clustering of the data described in Section 4.3, since any clustering conducted on a train set containing noise for the additive FDE is bound to form new clusters based on the appearance of noise. As seen in Table 3 the performance on this original clustering was increased with approximately 5% when the noise models were added, which equivalently removed about 25% of the recognition errors. This is very promising and shows the power of the additivity property of the method presented in this paper. By just adding two prototypes to the database, a large part of the samples which corresponded to digits with noise, were handled. A graphical example of an recognition error corrected in this way is shown in Figure 3.

Table 3: A comparison of FDE-SCR on original clustering (as in Section 4.3) with additive recognition with a Recognition Graph (RG)

Method	Prototypes	Max Compared	Top-1
RG	170	N/A	83.57%
FDE-SCR	168	24	78.47%

5 Conclusions

This paper presents a novel additive single character recognition method. The additivity property as defined in this paper ensures that the single character recognition method can be easily extended to treat cursive word recognition. However, the additivity property also poses a problem when comparing the novel method to existent methods since it does not perform very well if trained on noisy data. Experiments with simplistic noise models has shown that the method is bound to perform a lot better if noise is treated independently and future research will focus on finding methods to remove the noise from a training set in such a way that the clustering and prototype extraction can focus on the actual shapes of the recognition targets.

The paper also proposes a novel segmental curve matching algorithm to employ in the additive recognition method as presented here. The algorithm has been validated on all single segment samples in the UNIPEN-1a database with better results than conventional methods.

References

[1] J. Alon, V. Athitsos, and S. Sclaroff. Online and offline character recognition using alignment to prototypes. In *Proc. of the 8th International Conference on Document Analysis and Recognition*, pages 839–843, 2005.

[2] C. Bahlmann and H. Burkhardt. The writer independent online handwriting recognition system *frog on hand* and cluster generative statistical dynamic time warping. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(3):299–310, March 2004.

[3] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project of on-line data exchange and recognizer benchmarks. In *Proc. 12th International Conference on Pattern Recognition*, pages 29–33, Jerusalem, Israel, October 1994.

[4] S. Jaeger, S. Manke, and A. Waibel. Npen++: An on-line handwriting recognition system. In *Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pages 249–260, sept 2000.

[5] J. Ma, Y. Zhao, and S. Ahalt. OSU SVM Classifier Matlab Toolbox (ver. 3.00). http://eewww.eng.ohiostate.-edu/maj/osu_svm/, 2002.

[6] H. Mitoma, S. Uchida, and H. Sakoe. Online character recognition based on elastic matching and quadratic discrimination. In *Proc. of the 8th International Conference on Document Analysis and Recognition*, pages 36–40, 2005.

[7] J. Oh. *An On-Line Handwriting Recognizer with Fisher Matching, Hypotheses Propagation Network and Context Constraint Models*. PhD thesis, New York University, 2001.

[8] R. Plamondon and S. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22(1):63–84, January 2000.

[9] M. Schenkel and I. Guyon. On-line cursive script recognition using time delay networks and hidden markov models. *Machine Vision and Applications*, 8:215–223, 1995.

[10] J. Sternby. Frame deformation energy matching of on-line handwritten characters. In *Proceedings of the 10th Iberoamerican Congress on Pattern Recognition*, pages 128–137, Havanna, Cuba, 2005.

[11] J. Sternby and C. Friberg. The recognition graph - language independent adaptable on-line cursive script recognition. In *Proc. of the 8th International Conference on Document Analysis and Recognition*, pages 14–18, 2005.