Pre-Processing the Dynamics of On-Line Handwriting Data, Feature Extraction and Recognition

Homayoon S.M. Beigi

IBM T.J. Watson Research Center P.O. Box 218 - Yorktown Heights, New York 10598 U.S.A.
EMail: beigi@watson.ibm.com

Abstract

Most of the dynamic information present in an on-line handwriting signal is often ignored by on-line handwriting recognition systems. It is shown here that the dynamic information is complementary to the shape information and may be used to improve the accuracy of the recognition system.

1 Introduction

This paper presents results of an effort to use the dynamic information present in an on-line handwriting signal. On-line handwriting recognition systems often ignore most of the dynamic information available in the signal. They commonly go to the extent of retaining the order of points being sampled and throwing away all other dynamic (speed) information through a resampling of some sort. [1], developed by our group at IBM Research, is one such system which through resampling of the original data points gets rid of some of the velocity information. In such paradigms, to be able to compare feature vectors stemming from different pieces of the writing, since they are solely based on the shape of the curve, some size normalization would have to be done. Normalization ensures that equal length segments are compared against each other. In the algorithm used by the system in [1], the writing is re-sampled to produce equi-distant points; then a segment of the writing with a fixed number of points (hence the same Euclidean Length) is used to produce the feature vector. This feature extraction technique is used for both training and decoding. With a system of this type, the characters should be formed at a nominal size to get an acceptable comparison. For this reason, a size normalization is very important before training or decoding. [2]

To improve the recognition results of our current system, [1], an effort was initiated to look at features which would be generated based on the dynamics of the handwriting data under the assumption that parts of the information obtained through these features would be complementary to those generated by the existing features (referred to here as *static features* for the purpose of distinction). It is hoped that the union of these features in the form of complementary codebooks will increase the overall accuracy of the handwriting recognition system. The system of interest recognizes unconstrained handwriting (any combination of cursive or discrete writing styles). Size normalization is not as important with these features; although, a velocity (time) normalization would be required.

2 Handwriting Model

The general equation of motion of most rigid-body systems, with the consideration of dynamics such as inertia, gravity, Coriolis, Centrifugal, and other forces may be written as follows:

$$T = M(\alpha)\ddot{\alpha} + C(\alpha, \dot{\alpha}) + F(\dot{\alpha}) + G(\alpha) + T_d \tag{1}$$

where, T is the vector of generalized forces supplied by the actuators (muscles), α is the vector of generalized coordinates, $M(\alpha)$ is the equivalent mass matrix, $C(\alpha, \dot{\alpha})$ is the vector of generalized forces due to Coriolis and Centrifugal accelerations, $F(\dot{\alpha})$ is the vector of generalized forces due to generalized viscous friction, $G(\alpha)$ is the vector of generalized forces due to gravity and other potential energy such as stiffness, and T_d is the vector of disturbances, friction, and other unmodeled forces. [3] In equation 1, all vectors are of dimension $nx1$ and $M(\alpha)$ has dimension nxn.

Please note that equation 1 may be linearized about a set of reference coordinates at each time instant; the non-linear equations of motion can then be approximated by a set of second order linear differential equations. If we consider the governing equations of motion for the human-hand motor control, used for handwriting, we may make further simplifications by ignoring all forces but the D'Alembert forces and spring stiffness. These approximations only hold true under the conditions that the speed of writing is within limits of the average writing speeds and that the hand moves in a two dimensional trajectory related to producing legible handwriting. Hollerbach, in his 1980 Doctoral Thesis [4], made even greater simplifications to this model by assuming a decoupled mass matrix. However, the mass and spring stiffness would still be functions of time even if we ignore all other forces. Another assumption which also holds very well is that the values of the mass and the spring stiffness are piecewise constant along a pen trajectory. These pieces along which the system parameters remain nearly constant are bordered at points of the extrema in the coordinates (x and y coordinates). These points coincide with zero velocity points for the corresponding coordinate, extrema of the dynamics (figure 1). Under certain assumptions such as minimum energy, minimum jerk, etc., these boundaries and the models may change. [4, 5]

Considering the discussed approximations, let us then assume that the differential equations for the handwriting generation process may be approximated by a two dimensional second order differential equation with linear time-invariant coefficients along a piece of writing between any two consecutive extreme positions in each coordinate (x and y), given by figure 1. Under these assumptions, the solution of the approximate differential equation, in the approximation region, would be of the usual sine and cosine form. In fact the velocity in each coordinate will also have the same form. For the sake of modeling the handwriting it is better to consider the velocity rather than the position for the apparent reasons of robustness to noise and pre-emphasis. Thus, the velocities in x and y directions under these very crude assumptions would be of the form given in equation 2.

$$\dot{x} = A_x cos(\omega_x t + \phi_x) + \bar{v}_x$$
$$\dot{y} = A_y cos(\omega_y t + \phi_y) + \bar{v}_y \tag{2}$$

where A_x, ω_x, ϕ_x and $\bar{v}_x$ are the amplitude, frequency, phase and mean velocity for the x direction. Also, A_y, ω_y, ϕ_y and $\bar{v}_y$ are the counterparts in the y direction. Figure 3 shows the plot of $\dot{y} - \bar{v}_y$ for the word *languages* written in script (see top of figure 9 for the word). In figure 3, the segmentation boundaries have been marked. As it is apparent from the figure, the form of a cosine with constant parameters is quite an acceptable approximation for each piece in the plot. This approximation, although not as good, is also acceptable for the x direction to a certain extent. One may at the first glance imagine that the phase does not play an important role, however, if the word is to be segmented in such a way as to use **either** $\dot{x} = 0$ **or** $\dot{y} = 0$, then the phase plays a very important role of synchronization. In fact in some cases a segment boundary is simply deleted just because it generates too small a window; in these cases, also, the phase is essential (figure 1).

3 Pre-Processing

In a practical sense, if segments of the writing are written very slowly, a couple of problems may arise. The first problem is the computation of segmentation points where $\dot{x}$ and $\dot{y}$ are nearly zero. The finite difference approximation of velocities in x and y assumes a Δt which is small in comparison to the nominal Δx or Δy. However, in slow speeds, since the Δt is a fixed number equal to the inverse of the sampling frequency, the velocity is not correctly estimated. To alleviate this problem, a time normalization is done to generate writing with the same nominal speed. Figures 4 through 7 show the plots of x and y versus time for four different samples of the character *a* being written by the same writer. Shape of the character is the same for all four samples. In these figures, the darker lines show the original points and the lighter lines show the points after time normalization. The *a* in figure 6 is an example written in a nominal speed (empirically determined). The sample of figure 4 was written in a close to nominal speed with certain variations and as it may be seen from the graph, after normalization, the plot looks a lot more like those in figure 6. The sample of figure 5 was written with a uniform speed, but in general much more slowly than the nominal speed of writing. After time normalization, again, the plots are very close to those shown in figure 6. The character of figure 5 is shown in figure 2. In addition to causing problems in velocity computation, slowly written characters also contain a fair amount of noise. The noise is reduced by using a 10^{th} order low-pass Yulewalk filter, with zero phase shift, on the signal and the results are shown in figure 2 in a lighter line. This filtering is done before proceeding with the time normalization. Finally, figure 7 shows the most complicated case, namely, when the velocity of writing changes within a single stroke. This often happens when a person is preoccupied by a thought while writing. An example is when the person puts the pen on the pad and meanwhile thinks of the statement to write. The application of this time normalization is not only important in regards to segmentation, but it also allows the approximation of all the mean velocities within the segments by the same mean velocity over the whole stroke. This is particularly useful for reducing the number of parameters to be saved for a follow up reconstruction.

4 Reconstruction and Recognition

The parameters of equation 2 may be estimated and used for several purposes including reconstruction (hence compression) and recognition of the writing. Figures 8 and 9 show sample reconstructions using the estimates of parameters for equation 2. Figure 8 also shows the location of segmentation points. These segmentation points may be further reduced by using the cusp point between two adjacent segmentation points (if present) and removing the extraneous point. This is also important in removing noisy parameters when used for the purpose of recognition. The fast match shape recognition techniques of [1], based on a degenerate single-state continuous density Hidden-Markov Model, were used to come up with some preliminary shape recognition results, using these estimated parameters. In these tests, the digits from 0 to 9, upper and lower case letters were trained and tested. The system was trained on a total of 2303 characters and tested on a total of 833 characters. The distribution of characters was not at all uniform and due to space restrictions is not presented here. The same data files were also used for training and testing of the fast-match version of the system with static features [1]. Figures 10 through 12 show the results of these runs. Based on these results, performances of the two systems are quite complementary. These results show ideal conditions for combination of the two codebooks. For example, digit 1 and characters I, J, j and n get zero accuracy with the static features and an average of 76% accuracy with the dynamic features. Likewise, F, K and X get zero percent accuracy with the dynamic features and an average of 50% accuracy with the static features. The overall character accuracy of the system using dynamic features is 47% where the overall character accuracy of the system using static features is 66%.

Part of the worse total character accuracy of the system using the dynamic features is due to the lack of context. The system using static features, uses the window edges (see figure 1) as its window centers and looks to the left and right of these centers for a fixed length. These windows are overlapping one another, hence capturing some local context. [1] In the dynamic feature case, only the information within a window (between two consecutive segmentation points) is used to generate features and therefore no local context is available. It is quite acceptable to assume that by using a multi-state Hidden Markov Model, for both cases, the two performances should converge each other. However, the important point to realize is the complementary nature of the two systems and the fact that they may be used together to produce better results. This experiment is being worked on at the present time and no results are available yet. The delay in obtaining such results is partly due to the different front ends (windowing paradigms). Some writer-dependent fast match results are also available for unconstrained handwriting using the dynamic features. In these results are obtained using a lexicon of 368 words related to computer applications was used. Writers 1 and 2 had word accuracies of 91.5% (184 / 201) and 75.8% (229 / 302) respectively. When the training files of these two writers were combined into one and trained and tested on the same test files, they showed word accuracies of 87.1% (175 / 201) and 70.9% (214 / 302) respectively.

5 Conclusions and Future Research

If the combination of training files is continued further, as discussed at the end of the previous section, to include many writers in the training part, a write-independent prototype set will be established. Preliminary attempts at doing this have shown that the features should be tranformed (engineered) to produce better generalization capability through some normalization. Normalization procedures presented in the Pre-Processing section of this paper have been considered to achieve such results. These normalization techniques have not been used in the results shown in the paper. This is an on-going effort and new results will be presented in the near future.

It is also notable that the correlation coefficient between the frequencies in the x and y direction is on the order of 0.7 which suggests that one of the two frequencies may be omitted for the sake of recognition without any large performance loss. In fact, the removal of the ω_x resulted in a slightly higher accuracy for both writers of the last section. This is due to the reduction of the dimension of the Gaussian mixtures used for representing the dynamic features [1] from 6 to 5 which makes the means and variances of these Gaussians more accurate given the fixed amount of training data. For this reason, experiments have been made to reduce the dynamic features further down to 4 without any loss of accuracy. These experiments upon full completion will be presented in future publications. Due to the different window definition for computing the features in the new system and the system of [1], the two codebooks are not easily combined. An attempt is being made to generate outputs from both of these fast match techniques (static and dynamic) and then pass a combination of words proposed by these systems to the detailed match scheme of [1]. This will be done after increasing the generalization capabilities of the dynamic features through some transformations being considered.

References

[1] Krishna S. Nathan, Homayoon S.M. Beigi, G. J. Clary, J. Subrahmonia and H. Maruyama, "Real Time On-Line Unconstrained Handwriting Recognition using Statistical Methods," ICASSP-95, Detroit, Michigan, May 8-12, 1995.

[2] Homayoon S.M. Beigi, Krishna Nathan, G. J. Clary, and J. Subrahmonia, "Size Normalization in On-Line Unconstrained Handwriting Recognition," ICIP Proc., Nov. 13-16, 1994.

[3] J. J. Craig, Adaptive Control of Mechanical Manipulators, Addison-Wesley, New York, 1987.

[4] John M. Hollerbach, "An Oscillation Theory of Handwriting," PhD Thesis, MIT, 1980.

[5] Adel M. Alimi and Réjean Plamondon, "Performance Analysis of Handwritten Strokes Generation Models," Pre-Proc. of IWFHRIII, Buffalo, New York, May 25-27, 1993, pp.272-283.

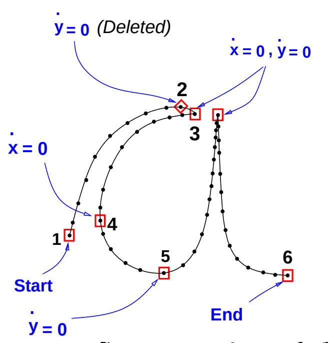

Figure 1: Segmentation of the character *a*

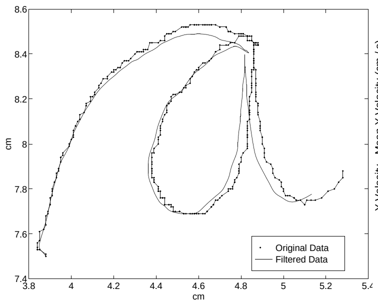

Figure 2: character *a* written very slowly

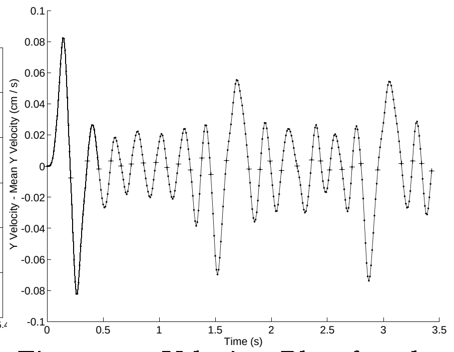

Figure 3: Velocity Plot for the Word *languages*

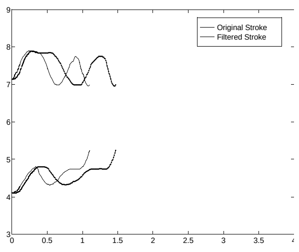

Figure 4: *a*: normal speed with small variations

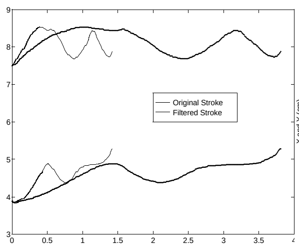

Figure 5: *a*: written very slowly

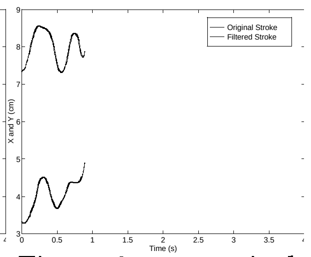

Figure 6: *a*: nominal speed

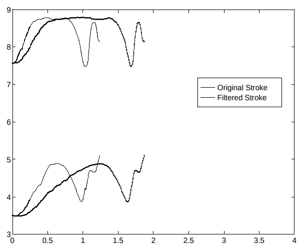

Figure 7: *a*: very slowly (first ligature); nominal speed (rest)

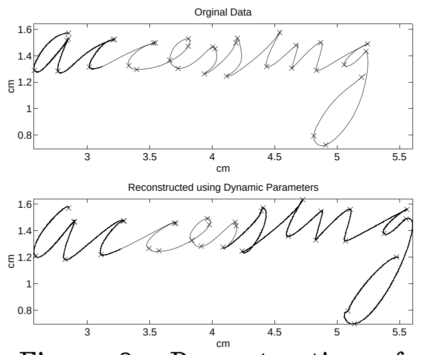

Figure 8: Reconstruction of *accessing* using Dynamic Parameters

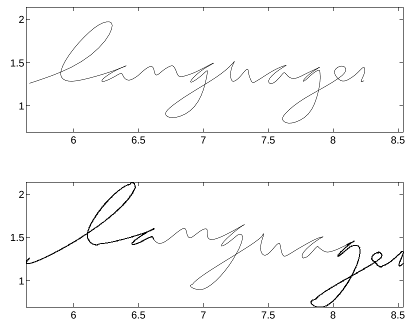
Figure 9: Reconstruction of *languages* using Dynamic Parameters

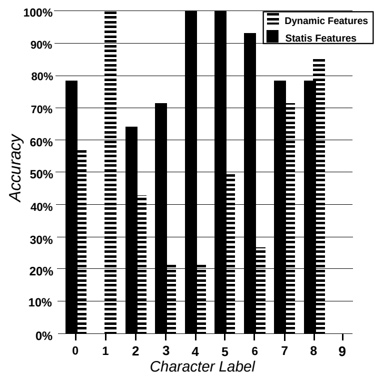

Figure 10: Accuracy Comparison for Digits

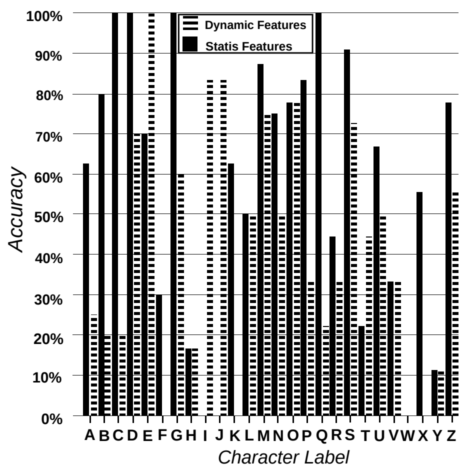

Figure 11: Upper Case Acc. Comparison

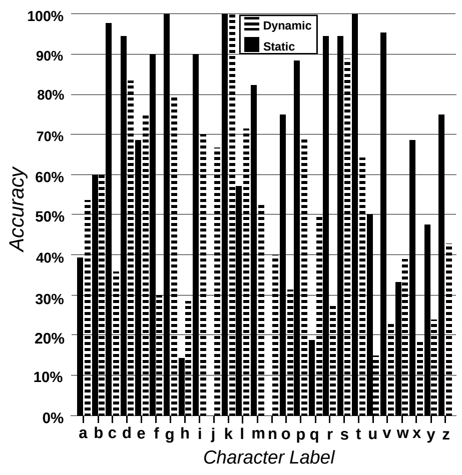

Figure 12: Lower Case Acc. Comparison