Representing and Characterizing Handwritten Mathematical Symbols through Succinct Functional Approximation

Bruce W. Char
Department of Computer Science
Drexel University, Philadelphia, PA 19104 USA
charbw@drexel.edu

Stephen M. Watt
Department of Computer Science
University of Western Ontario
London, Ontario N6A 5B7 Canada
watt@orcca.on.ca

Abstract

We model on-line ink traces for a set of 219 symbols to "best fit" low-degree polynomial series. Using a collection of mathematical writing samples, we find that in many cases this provides a succinct way to model the stylus movements of actual test users. Furthermore, even without further similarity-processing, the polynomial coefficients from the writing samples form clusters which often contain the same character as written by different users. We find this style of characterization to be an attractive tool due to the suitability of the representation to computation and mathematical analysis.

1 Introduction

On-line handwriting recognition deals with ink traces, typically given as sequences of (x, y) coordinates at various time values, as collected from some electromagnetic or optical digitizing device. For example, the emerging InkML[3] standard presents stylus input as a series of strokes, each stroke containing (possibly with other values) a sequence of tuples $(x_0, y_0, t_0), (x_1, y_1, t_1) \ldots (x_n, y_n, t_n)$, where $x_i, y_i, t_i \in \mathcal{R}, 0 \leq i \leq n,$ and $t_0 < t_1 < \cdots < t_n$. The first two elements of each tuple usually represent the two dimensional position of the pen tip at a particular moment in time. The time value may be given as a value in the tuple or implicitly, implied by a sampling rate. In general, we cannot assume that the time values t_i are equally spaced.

On-line recognition software usually processes input ink strokes, for example to scale them, smooth them or resample them (replacing actual points by a desired number of interpolating points). The representation of the digital ink remains essentially the same, however, as a sequence of data points along ink traces.

In processing stylus input as handwriting, one could disregard the time sequence information and proceed with an optical character recognition (OCR) processing of the (x, y) information as a set of positions with no particular ordering. However, handwriting recognition software can benefit from the time sequencing data, as with elastic matching of the input sequence of points to the sequences of model characters (see e.g. [13]), or feature extraction such as the calculation of angles[14], finding loops and curves[2], etc.

We are motivated by the problem of writer-independent recognition of mathematical handwriting, where the usual approaches to character recognition are not as successful as in ordinary text. In this context, we have been led to examine some old questions afresh and, in particular, to examine recognition methods based on alternative ink representations.

Recognition for handwritten mathematical expressions is more complicated than for ordinary handwriting because of the two-dimensional nature of the input, because there are more symbols involved, and because the greater difficulty of using contextual information (e.g. harder to provide a dictionary of words or phrases against which the input is probably coming from). Since past techniques have not provided a complete solution, the door is still open for additional low-cost techniques that will help.

We explore the idea of modeling the sequence of positions taken by a stylus while writing a mathematical symbol to "best fit" Chebyshev polynomials. Using a collection of writing samples of mathematical symbols, we find that in many cases it provides a succinct way to model the stylus movements of actual test users. The principal contributions of this paper are:

- We find that modeling $x(t)$ and $y(t)$ as Chebyshev series accurately captures the shape of handwritten mathematical characters using few parameters.
- Even without further similarity-processing, we find the polynomial coefficients from the writing samples form clusters which often contain the same character as written by different test users.

The paper is organized as follows: Section 2 describes how to model traces by expressing $x(t)$ and $y(t)$ as a series

in an arbitrary functional basis, and then specifically as a a Chebyshev series. Section 3 describes our collection of handwritten mathematical symbols. Section 4 describes the quality of the models produced. Section 5 presents some preliminary results on character classes obtained by clustering models based on their series coefficients. Section 6 relates our approach to previous work, presents conclusions, and directions for further investigation.

2 Modeling data by approximation with a functional basis

A well-known technique is to analyze an arbitrary function $f : \mathcal{R} \rightarrow \mathcal{R}$ by writing f as a linear combination of elements of set of basis functions $B = \{b_i : \mathcal{R} \rightarrow \mathcal{R}, i = 0, 1, \ldots\}$:

$$f(t) = \sum_{i=0}^{\infty} c_i b_i(t), \qquad c_i \in \mathcal{R}, b_i \in B \tag{1}$$

Often the family of functions in B are designed so as to be orthonormal: $\langle b_i, b_j \rangle = \int_I b_i(t) b_j(t) w(t) \mathrm{d}t = \delta_{ij}$, integrating with weight $w(t)$ over a suitably defined interval $I = [R_1, R_2]$ so that the series (1) is unique. For such a basis, when $w(t) = 1$ then $c_i = \langle f, b_i \rangle$.

One can gain insight into f from knowledge of the family of functions in B and the values of coefficients c_i. If the series defined by (1) closely approximates f in a few terms, then behavior of f can largely be understood knowing c_i and b_i of these terms.

For our analysis, we assume that handwriting traces are provided as sequences of (x_i, y_i, t_i) tuples that have been normalized so $t_0 = 0$ and $t_n = 1$. For basis functions we use Chebyshev polynomials of the first kind, defined by $T_n(t) \equiv \cos(n \arccos t)$ and orthogonal for the inner product $\langle f, g \rangle = \int_{-1}^{1} f(t) g(t) / \sqrt{1 - t^2} \mathrm{d}t$. We have also explored the use of Bernstein polynomials as basis functions, but have not found them as satisfactory for succinct modeling of our handwriting samples.

We treat our empirical data traces as functions on $[0, 1]$ by linear interpolation of the x and y coordinates: $X(t) = x_i + \frac{x_{i+1} - x_i}{t_{i+1} - t_i}(t - t_i)$, for $t_i \leq t < t_{i+1}, 0 \leq i < n$, and similarly for $Y(t)$. We then develop the *Chebyshev series* for X and Y

$$X(t) = \sum_{i=0}^{\infty} \alpha_i T_i(t), \quad Y(t) = \sum_{i=0}^{\infty} \beta_i T_i(t). \tag{2}$$

Well-known techniques exist for computing the coefficients of a Chebyshev series for a function [4] and are implemented in commonly available mathematical software libraries, e.g. [5, 9].

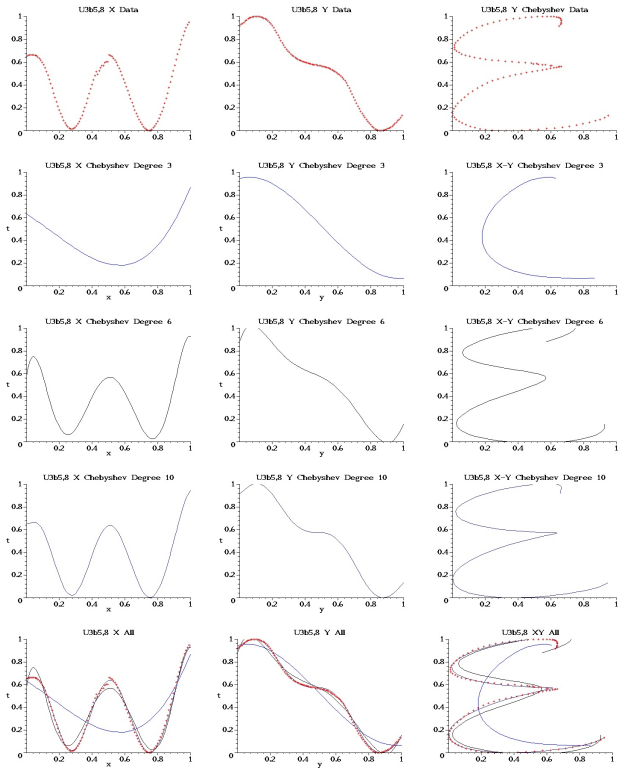

Figure 1. The x, y and combined traces of ϵ: Data, Degree 3, 6, 10 Chebyshev fits, Superimposed.

3 The ORCCA collection of mathematical handwriting samples

Our samples were drawn from a collection of mathematical handwriting assembled at the Ontario Research Centre for Computer Algebra (ORCCA). The collection was taken from $U = 9$ test users, who were asked to provide handwritten samples of $M = 239$ different mathematical expressions. Most of these expressions corresponded to a particular Unicode symbol drawn with one or more strokes. All test users provided samples to most of the symbols, but none provided samples for all characters. (It was considered undesirable to have users write symbols unfamiliar to them.) There were $N_1 = 1878$ different samples overall. Of these, there were $N_2 = 908$ samples that consisted of one stroke for the entire character. To simplify processing, samples were normalized in the following ways:

1. Data set positions equally spaced in time.
2. $0 \leq x_i, y_i, t_i \leq 1$ in every sample, by dividing the raw data values by the appropriate trace minimum and maximum. This had the effect of having the normalized trace have domain and range within $[0, 1]$, with every character having either its x or y values extend to the full range of $[0, 1]$.

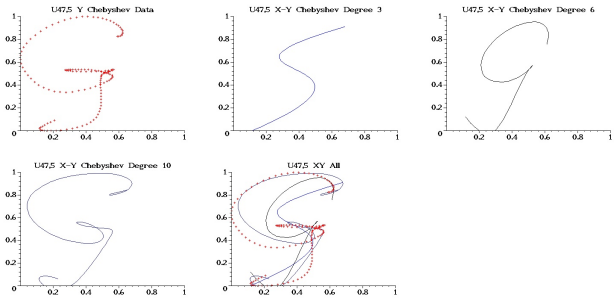

Figure 2. The (x, y) trace of $\mathcal{G}$: Data, Degree 3, 6, 10 Chebyshev fits, Superimposed.

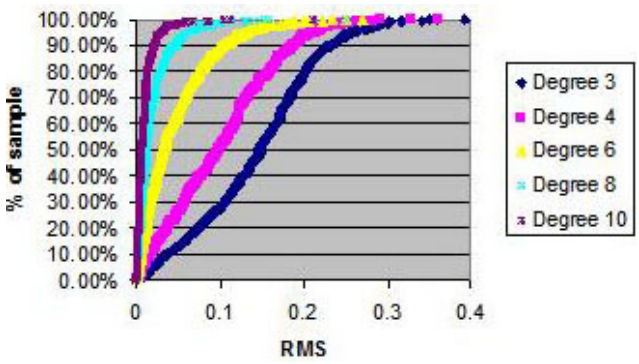

Figure 3. RMS error Chebyshev fits by degree

4 Quality of series approximation

Initially, we computed series approximations to all 908 writing samples where the entire character was written with one stroke. Figure 1 shows fits of $x(t)$ and $y(t)$ for a typical sample with Chebyshev approximations of degrees 3, 6 and 10. As can be seen from Figures 1 and 2, the polynomials do a reasonable job of approximating smoothly drawn characters, even in some cases where there are cusps or loops. They reproduce less faithfully writing samples with sharper changes, although even in that case the result is quite suggestive of the character being written.

To explore the feasibility and desirability of Chebyshev fitting, we computed approximations for the N_2 samples which consisted of one stroke. For each sample fitted, we computed the root mean square of the deviation between the sample and the corresponding Chebyshev value:

$$\mathrm{RMS}(\phi) = \sqrt{\sum_{i=0}^{n} \frac{\|\phi(t_i) - (x_i, y_i)\|_2{}^2}{n+1}} \qquad (3)$$

for a result function $\phi : t \rightarrow \mathcal{R}^2$ for a particular sample.Due to normalization, the range of ϕ and x_i and y_i are in $[0, 1]$, so the range of the RMS function is $[0, \sqrt{2}]$. Table 1 and Figure 3 show the distribution of RMS for the samples.

Degree	$< 90\%$	$< 95\%$	$< 99\%$
3	.230	.256	.302
4	.188	.211	.254
6	.103	.131	.184
8	.045	.060	.102
10	.021	.029	.059

Table 1. RMS cutoffs by Chebyshev degree

5 Clustering and Chebyshev coefficients

The Chebyshev polynomials were computed for the N_2 one stroke handwriting samples, which covered $C = 219$ distinct characters. We explored the clustering that occurred for various numbers of clusters using an efficient implementation of the k-means algorithm [7, 8, 10], using the Euclidean metric, $\delta(p, q) = \sqrt{\sum_{i=0}^{d}(p_i - q_i)^2}$ as the distance measure. The average distances for the various degrees is shown in Table 2, indicating again the conclusion that there is little difference between results for polynomials beyond degree 6. We show in Figures 4–6 elements of a few clusters when 260 clusters are specified for the k-means algorithm. Out of the $N = 906$ samples, 870 were in clusters of size ≥ 2 and 740 were in clusters of size ≥ 3.

We have begun experimentation with clustering based on polynomials coefficients based on simple variations on sample normalization. One can observe in the results of Figure 4 that "integral", "slash" and "comma" were being classified together. We determined that this was due to our decision to normalize all characters to the same scale regardless of the size that they were originally written, as described in step 2 of section 3. An alternative would be to normalize all samples from a user according to the maximum span over all samples of that user. Since the fit coefficients are sensitive to scale, scaling in this alternative way causes similar-shaped samples of a character to fall into different clusters due to differences in character size. We have found

Degree	Mean dist. between polys
3	0.689607
4	0.718386
6	0.745591
8	0.753625
10	0.755495

Table 2. Degree vs average distance between fit polynomials

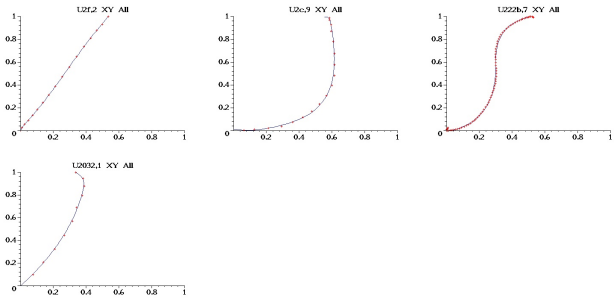

Figure 4. Elements of Cluster 69 for k-means run specifying 260 clusters. The cluster has 9 elements: four different test users' versions of '/' (slash), two each of ',' (comma) and '$\int$' and one of '′' (prime). The maximum distance to the cluster center is .197.

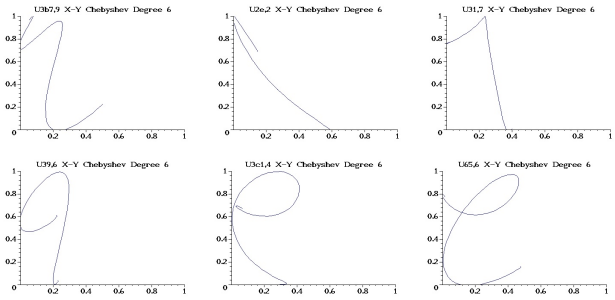

Figure 5. Elements of Cluster 5 for k-means run specifying 260 clusters. The cluster has 8 elements: three versions of 'η', and one each of 5'.', '1', '9', 'ρ' and 'e'. The maximum distance to the cluster center is .063

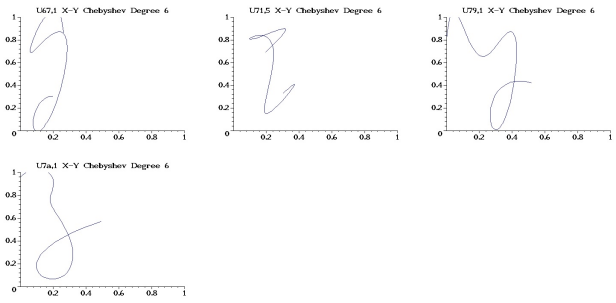

Figure 6. Elements Cluster 12 for k-means run specifying 260 clusters. The cluster has 7 elements: 4 versions of 'g', and one each of 'z', 'y' and 'q'. The maximum distance to the cluster center is .048.

that clustering using the concatenation of coefficients from the uniformly- and user-normalized scalings leads to better results in some cases. For example, k-means clustering using 260 clusters with the concatenated coefficients leads to a cluster with 6 occurrences of "integral" (the two from "Cluster 69", plus four more), and only one other character, a "slash" from the previous clustering. A cluster with the same six samples of "integral", and another cluster with the same 4 samples of "comma" occurred using both approaches. While this approach seems to do a better job of discriminating and clustering some kinds of characters, there continue to be clusters with a confluence of different characters as written by different subjects.

6 Comparison with prior work, Conclusions and Future directions

Some prior work has looked at representation of data through other functions, notably through Fourier and wavelet analysis. For example, Sandaresan *et. al.* used Fourier series and wavelets to model handwritten Tamil using Fourier transform coefficients and wavelets functions [11]. They claimed that wavelets were better than Fourier series for capturing the localized differences that help differentiate between some pairs of Tamil characters. Wavelet functional signatures have been explored in an OCR context for handwritten Chinese characters [6] and for printed alphanumeric characters [12, 15]. In [12], a different coordinate system was used to make the functional signatures invariant under rotation of the data.

In [1], Hermite polynomial transforms were applied to extended handwriting samples to develop a handwriting"textures" classification that could be used to identify a sample as being from a particular person. Zhang, Bui and Suen [16] used functional classification (based on wavelets) as one part of a hybrid recognizer for handwritten numerals. The architecture they propose would also also be an appropriate way of combining the strengths of Chebyshev characterization with other techniques for recognition.

Initially, our choice of Chebyshev polynomials as the basis functions was based on computational convenience on computing series. However, our analysis suggests that "low frequency changes" which can be approximated by low-degree Chebyshev series is a fundamental feature of many handwritten samples of mathematical symbols. This suggests that it or a similar low-cost mathematical representation of the data can be useful in handwriting recognition in this domain.

One clear conclusion of this work is that representation of ink traces using functional bases is an interesting and useful direction. We find that clusters of an appropriate radius often include a number of samples of the same character as given by different test users. However, they can

also include similar but different characters. With minimal effort we have been able to automatically place characters in clusters of similar shape. This may be used for quick pre-classification, used in a multi-classifier setting. As our experiment with the concatenated coefficients in section 5 suggests, different normalization or additional filters easily computed from the original data could lead to more powerful classification results, so is worthy of further investigation.

The more tantalizing aspect of functional representation in general, and polynomial representation in particular, is the conceptual economy it brings to the analysis of the curve traces. Since low degree implicit polynomial representation gives high quality approximation to traces, a spectrum of mathematical tools may be brought to bear on the polynomial functions. For example, the identification of inflection points and cusps is easy in this representation, while in the data point representation involves arbitrary parameters in inspecting neighboring point sets. A secondary property of this representation is its compactness, reducing the number of parameters to describe a trace by up to an order of magnitude. All of the analysis that would be done on the point representation, and more, can be done on the functional representation.

Given the evident utility of this representation, there are several questions that merit further investigation. These include: parameterization by arc-length rather than time, inclusion of time and force as coordinates for fitting, choice of coordinate system (polar *vs* Cartesian *vs* custom coordinates to preserve conformal invariance), choice of basis functions and scaling strategies. In addition, generalization to multi-stroke characters should be investigated as should be treatment of multi-character symbols (*e.g.* `sin`, `log`, *etc.*)

7 Acknowledgments

This work was conducted with the support of Microsoft Canada, the National Science Foundation grants REC-0325872 and CCF-0325685, the NSERC Discovery Grants Program, Drexel University and the University of Western Ontario. We also wish to acknowledge the assistance of Elena Smirnova, Laurentiu Dragan, and Jeliazko Polihronov of ORCCA (University of Western Ontario), and Trip Denton of the Applied Algorithmics Laboratory (Drexel).

References

[1] Stephane Bres, Veronique Eglin, and C. Volpihac-Auger. Evaluation of Handwriting Similarities Using Hermite Transform. In *Proc. 10th Intl. Workshop on Frontiers in Handwriting Recognition*, October 2006. `http://hal.inria.fr/view_by_stamp.php?label=IWFHR10&langue=en&action_todo=view&id=inria-00108345&version=1`.

[2] K.F. Chan and D.Y. Yeung. Elastic structural matching for online handwritten alphanumeric character recognition. Technical report, Department of Computer Science, Hong Kong University of Science and Technology, March 1998.

[3] Y-M. Chee, M. Froumentin, and S.M. Watt (eds). Ink Markup Language (InkML). `http://www.w3.org/TR/InkML/`, October 2006.

[4] C.W. Clenshaw. Chebyshev series for mathematical functions. *Nat. Phys. Lab. Math. Tables*, 5, 1962.

[5] Brian Gough, Jim Davies, James Theiler, Mark Galassi, Gerard Jungman, Michael Booth, and Fabrice Rossi, editors. *GNU Scientific Library Reference Manual*. Bristol, 2nd edition, 2006.

[6] Lei Huang and Xiao Huang. Multiresolution recognition of offline handwritten Chinese characters with wavelet transform. In *ICDAR '01: Proc. 4th Intl. Conf. Doc. Analysis and Recog.*, page 0631. IEEE Computer Society, 2001.

[7] T. Kanungo, D.M. Mount, N. Netanyahu, C. Paitko, R. Silverman, and A.Y. Wu. A local search approximation algorithm for k-means clustering. In *Proc. 18th ACM Symposium on Comp. Geometry*, pages 10–18, 2002.

[8] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Chyristine D. Piatko, Ruther Silverman, and Angela Y. Wu. Kmlocal. `http://www.cs.umd.edu/~mount/Projects/KMeans/`, October 2005. Version 1.7.1.

[9] Maplesoft. *Maple User Manual*. Maplesoft, 2005.

[10] David M. Mount. `http://www.cs.umd.edu/~mount/Projects/KMeans/kmlocal-doc.pdf`, August 2005.

[11] C. S. Sundaresan and S. S. Keerthi. A study of representations for pen based handwriting recognition of tamil characters. In *ICDAR '99: Proc. 5th Intl. Conf. on Doc. Analysis and Recog.*, page 422, Washington, DC, USA, 1999. IEEE Computer Soc.

[12] Yuan Y. Tang, Bing F. Li, Hong Ma, Jiming Liu, C.H. Leung, and Ching Y. Suen. A novel approach to optical character recognition based on ring-projection-wavelet-fractal signatures. In *Proc. 13th Intl. Conf. on Pattern Recognition*, volume 2, pages 325–329, 1996.

[13] C. C. Tappert, C. Y. Suen, and T. Wakahara. The state of the art in online handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 12(8):787–808, 1990.

[14] H.-J. Winkler. HMM-based Handwritten Symbol Recognition using On-line and Off-line Features. In *ICASSP 96*, pages 3438–3441, 1996.

[15] Patrick Wunsch and Andrew F. Laine. Wavelet descriptors for multiresolution recognition of handprinted characters. *Pattern Recognition*, 28(8):1237–1249, August 1995.

[16] P. Zhang, T.D. Bui, and C.Y. Suen. Hybrid feature extraction and feature selection for improving recognition accuracy of handwritten numerals. In *ICDAR '05: Proc. 8th Intl. Conf. Doc. Analysis and Recog.*, volume 1, pages 136–140, 2005.

Errata

22 May 2008 Section 2, page 2, corrected

> "orthonormal for the inner product $\langle f, g\rangle = \int_{-1}^{1} f(t)g(t)\mathrm{d}t$"
>
> with
>
> "orthogonal for the inner product $\langle f, g\rangle = \int_{-1}^{1} f(t)g(t)/\sqrt{1-t^2}\mathrm{d}t$".