Digital Ink Compression via Functional Approximation

Vadim Mazalov and Stephen M. Watt
University of Western Ontario
London, Ontario, Canada
vmazalov@csd.uwo.ca, watt@csd.uwo.ca

Abstract

Representing digital ink traces as points in a function space has proven useful for online recognition. Ink trace coordinates or their integral invariants are written as parametric functions and approximated by truncated orthogonal series. This representation captures the shape of the ink traces with a small number of coefficients in a form quite compact and independent of device resolution, and various geometric techniques may be employed for recognition. The simplicity and high performance of this method lead us to ask whether the same idea can be applied to another important aspect in online handwriting – the compression of digital ink strokes. We have investigated Chebyshev, Legendre and Legendre-Sobolev orthogonal polynomial bases as well as Fourier series and have found that Chebyshev representation is the most suitable apparatus for compressing digital curves. We obtain compression rates of 30× to 50× and have the added benefit that the Legendre-Sobolev form, used for recognition, may be obtained by a single linear transformation.

1. Introduction

Digital ink has become an important medium for communication. Each year more devices support digital ink in some form, for capture, processing or recognition. These devices have a wide range of form factors and resources, from small hand-held devices to digital whiteboards. These devices are used in various configurations, individually, tethered for a single user, or in multi-party collaboration. Various vendor-specific binary formats are used to represent digital ink, and there is the vendor-neutral XML format InkML. With this increased use of digital ink, its efficient handling has become increasingly important. Small devices need to be able to handle it efficiently. On more powerful devices ink-handling applications may need to store a significant amount of model data to support recognition.

In our work on symbol recognition we have found it useful to represent ink strokes in a functional form, as coefficients of truncated orthogonal series [5, 7]. This form, described in more detail below, has the property that the shapes of the curves are given quite succinctly. It is natural to ask how this form may be used for compression. This is the subject of this paper. A consequence of this work is that we almost directly do recognition on compressed ink. We note that orthogonal polynomial methods have been used to compress speech and other signals, e.g. [1, 10, 15].

We take the view that lossless compression at time of ink capture is not a meaningful objective as each ink capture device has a resolution limit and sampling accuracy. So long as the reconstructed curve lies within these limits, lossy and lossless compression are equivalent. For our own applications involving recognition, lossless compression has no benefit. Small perturbations in strokes give symbols that a human reader would recognize as the same.

This paper studies how functional approximation techniques may be used for digital ink compression. We compare the compression rates obtained using a variety of functional bases, and find that a quite satisfactory compression rate may be achieved. Indeed, we see that a curve, compressed with functional approximation at device resolution, occupies about twice less space than the lossless compression with second differences.

The rest of the paper is organized as follows. Section 2 provides the necessary preliminaries: we review applicable ink formats, related work and the basics of functional approximation. Section 3 explains the problem we solve and gives an example of approximation of curves with different error limits. Section 4 describes the compression method. The experiments to evaluate the compression method are described in Section 5. Compression results for representing coefficients in text and binary formats are given in Sections 5.2 and 5.3 respectively. Section 6 concludes the paper.

2. Preliminaries

2.1 Ink Representation

A variety of digital ink standards are in use today. Among these one can mention vendor-specific or special-purpose formats: Jot [13], Unipen [8], Ink Serialized Format (ISF) [12] or Scalable Vector Graphics (SVG) [4]. In 2003, W3C introduced a first public draft of an XML-based markup language for digital trace description, InkML. This has evolved to the current standard definition in 2010 [14]. InkML has received an increasing attention due to its vendor neutrality and XML base. In the general case, a trace is given in InkML as a sequence of multidimensional points. Each coordinate gives the value of a particular channel at that point. Predefined channels include x and y coordinates, pressure and various angles.

2.2 Other Ink Compression Methods

A lossy algorithm was presented in [11], based on stroke simplification. It suggests to eliminate excessive points, forming a skeleton of the original curve. The algorithm is based on iterative computation of chordal deviation (the distance between the original curve and the simplified one) and elimination of the point with the minimum distance until the minimum distance becomes larger than a predefined threshold. Intuitively, such simplification may make curves jagged. Indeed, authors address this issue and propose to interpolate strokes on the decompression stage with Hermite splines. However, since the interpolated values are obtained from a subset of the original points, noticeable approximation error is unavoidable.

A lossless compression scheme was proposed in [12] and similarly in [3]. The algorithm computes the second order differences of data items in each data channel. The sequence of second differences is proposed to have low variance and, therefore, be suitable for an entropy encoding algorithm. We show further on in the paper that compression of our algorithm is considerably better, even though it has a disadvantage of being lossy.

Another method, proposed in [3], is a so-called "substantially lossless", which allows the compression error's magnitude to be not greater than the sampling error's magnitude. In this approach, the original curve is split into segments and each segment is represented by some predefined shape, such as a polygon, ellipse, rectangle or Bezier curve. It is however not mentioned how to obtain the shapes from a curve and what compression this approach gives.

In contrast to the above methods, we look at piecewise functional approximation of a curve. Our approach allows flexible approximation with a desired level of precision by increasing the degree of approximation or decreasing the segment arc length. In addition, it yields high compression, as shown in the experimental part of the paper.

2.3 Orthogonal Bases

We have used polynomial bases extensively in our previous work. For details see, *e.g.*, [7] and works cited there. Here we summarize a few basic facts.

Let B_i be a set of basis functions orthogonal with respect to the inner product $\langle f, g\rangle = \int_a^b f(\lambda)g(\lambda)w(\lambda)\mathrm{d}\lambda$ on the interval $[a, b]$ with weight function w. If we are given an inner product, we may compute an orthogonal polynomial basis by Gram-Schmidt orthogonalization of the monomials $1, \lambda, \lambda^2, ...$

Let $f(\lambda)$ be an arbitrary continuous real-valued function on $[a, b]$. According to the Weierstrass approximation theorem, $f(\lambda)$ can be uniformly approximated to a desired degree of accuracy by a polynomial. In practice, orthogonal polynomials are usually taken as the basis for approximation, since they are uncorrelated and allow fast computation of coefficients. The expression of $f(\lambda)$ in the orthogonal basis B_i is

$$f(\lambda) = \sum_{i=0}^{\infty} c_i B_i(\lambda)$$

where the c_i are real coefficients. The coefficients of the functional expansion are found as

$$c_i = \langle f, B_i\rangle / h_{ii}, \qquad h_{ij} = \langle B_i, B_j\rangle .$$

For our purposes, the function $f(\lambda)$ is given by a discrete set of points, assembled in a stroke.

The interval of orthogonality in the definition of most of the classical orthogonal series is $\tau \in [-1, 1]$. Therefore a mapping to a more general interval is required. If $f(\lambda)$ is defined on $[a, b]$, coefficients of the mapping function can be obtained by taking $\lambda = (b-a)\tau/2 + (a+b)/2$

$$\begin{aligned}\hat{c}_i &= \frac{1}{h_{ii}} \int_{-1}^{1} \hat{f}(\tau)B_i(\tau)w(\tau)d\tau \\ &= K_i \int_a^b f(\lambda)B_i(\frac{2\lambda - a - b}{b-a})w(\frac{2\lambda - a - b}{b-a})d\lambda\end{aligned}$$

where $K_i = \frac{2}{h_{ii}(b-a)}$. Then coefficients of the degree d approximation of the original stroke can be found as

$$f(\lambda) \approx \sum_{i=0}^{d} \hat{c}_i B_i(\frac{2\lambda - a - b}{b-a}).$$

2.4 Bases for Approximation

We wish to determine which bases will be suitable for compression. We have investigated three families of orthogonal polynomials with useful properties and have included Fourier series for comparison.

Chebyshev polynomials of the first kind, defined as $T_n(\lambda) = \cos(n \arccos \lambda)$, have weight function $w(\lambda) = \frac{1}{\sqrt{1-\lambda^2}}$ and are used in numerical approximation for their property of minimizing the maximum error. In [2] it was reported that Chebyshev polynomials are suitable for succinct approximation of strokes and perform better than Bernstein polynomials.

Legendre polynomials are defined as

$$P_n(t) = \frac{1}{2^n n!} \frac{d^n}{dt^n}(t^2 - 1)^n$$

and have weight function $w(\lambda) = 1$.

Legendre-Sobolev polynomials are constructed by applying the Gram-Schmidt orthogonalization to the monomial basis $\{\lambda^i\}$ using the inner product

$$\langle f, g \rangle = \int_a^b f(\lambda)g(\lambda)d\lambda + \mu \int_a^b f'(\lambda)g'(\lambda)d\lambda$$

where $\mu = 1/8$ as described in [7].

A property of the Legendre and Legendre-Sobolev orthogonal bases, as applied to online stroke modeling, is the ability to recover a curve from moments computed in real time, while the stroke is being written. The coefficients of the stroke may then be calculated on pen-up in constant time depending only on the degree of approximation [6].

Fourier series on $[-L, L]$ are provided for comparison, since we are not restricted in our selection of approximation basis.

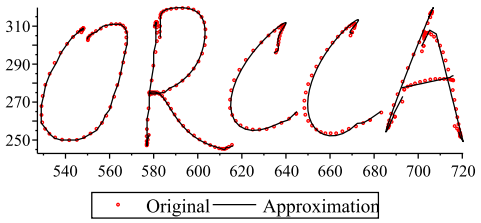
$$f(x) \approx \frac{\alpha_0}{2} + \sum_{n=1}^{d} (\alpha_n \cos(\frac{n\pi x}{L}) + \beta_n \sin(\frac{n\pi x}{L}))$$

where

$$\begin{bmatrix} \alpha_n \\ \beta_n \end{bmatrix} = \frac{1}{2L} \int_{-L}^{L} f(x) \begin{bmatrix} \cos \\ \sin \end{bmatrix} (\frac{n\pi x}{L}) dx.$$

3. Problem Statement

The question asked is whether it is feasible to apply the theory of functional approximation to describe a stroke up to some given threshold of the maximal pointwise error and root mean square error. If so, what is the compression one could expect as the result of such approximation?



	O	R	C	C	A
Max. pt-wise err., %	1	2	3	4	5
RMSE, %	0.33	0.67	1	1.33	1.67

Table 1. Different approximation thresholds.

We have empirically investigated different approaches to obtain the minimal overall size of coefficients of an approximation that satisfies the given error constraints. We have considered compression of handwritten regular text, since it commonly occurs in pen-based computing and incorporates different kinds of patterns. An example word and its approximation with different thresholds are shown in Table 1 and the corresponding figure. We have observe that limiting the maximum error also limits the root mean square error, but not vice versa. Therefore, in our experiments we have limited the maximum error. To measure the quality of approximation independently of application and device, we have computed errors (and other lengths) as a fraction of the height of the characters in a stroke.

4. Algorithms

4.1 Overview

At a high level, our compression method takes the following steps for each stroke:

1. Segment the stroke using one of the methods described below. Ensure the segments overlap by an amount at segmentation points.
2. For each segment, compute the orthogonal series coefficients for each coordinate function (*e.g.* x, y, p, where p is pressure).
3. Compress the stream of coefficients.

To reconstruct a stroke, the process is reversed:

1. Decompress the coefficient stream to obtain the curve segments.
2. Blend the curves on the overlaps to obtain the piecewise coordinate functions.
3. Obtain traces by evaluating the coordinate functions with the desired sample frequency.

On a given segment, the series coefficients are computed by numerical integration of the required inner

products. The cost to compute the compression is linear in the number of trace sample points and in the number of coefficient size/approximation degree combinations allowed.

To obtain a more compact form for the coefficient stream, it may be compressed with a deflation tool. In the experiments below we use gzip, which implements a combination of LZ77 [16] and Huffman coding [9].This is for convenience only — a more specialized method would be used in a production setting.

4.2 Parameterization Choice

We tested two used choices for curve parameterization widely used in pen-based computing: time and arc length. We observed that parameterization by time, while being easier to compute, also gives better compression. Comparison of the results is presented in Figure 2 for approximation with Chebyshev polynomials with integer coefficients.

4.3 Segmentation

We cannot expect long, complex strokes to be well approximated by low degree polynomials. Instead of varying the degree to suit any stroke, we segment strokes into parts that we can separately approximate. We have explored the three methods to segment traces, described here.

Fixed Degree Segmentation We fix the degree of the approximating functions. Intervals of approximation are constructed to allow the maximal length within the given error threshold. If the available interval can be approximated with a lower degree (*e.g.* the end of the curve has been reached), it is handled appropriately.

Fixed Length Segmentation We fix the length of intervals and approximate each interval with the minimal degree possible, but not greater than 20 (to keep the algorithm computationally feasible).

Adaptive Segmentation The most comprehensive variant is to fix a maximum permissible degree and maximum permissible coefficient size (digits for text, bits for binary), and to perform fixed degree segmentation for each combination. Then the combination of degree and coefficient size that gives the smallest resulting total size is selected. The degree and coefficient size are saved together with the coefficient data.

4.4 Segment Blending

If we allow a large error threshold (*e.g.* 4%), then it becomes possible to notice naïve segmentation because we do not match derivatives at the segmentation points. This can be observed in the Table 1. To make the stroke smooth, and to improve the approximation, we blend the transition from one piece to another by overlapping the segments slightly and transitioning linearly from one segment to the next on the overlap. Therefore, the approximation is given in segments, f_j, and takes form

$$f(\lambda) = \sum_{j=1}^{N} W_j(\lambda) f_j(\lambda) \approx \sum_{j=1}^{N} W_j(\lambda) \sum_{i=0}^{d} c_{ij} P_i(\lambda)$$

with the weight function

$$W_j(\lambda) = \begin{cases} 0, & \lambda \le \lambda_j - a \\ \frac{\lambda - (\lambda_j - a)}{a}, & \lambda_j - a < \lambda \le \lambda_j \\ 1, & \lambda_j < \lambda \le \lambda_{j+1} - a \\ \frac{-\lambda + \lambda_{j+1}}{a}, & \lambda_{j+1} - a < \lambda \le \lambda_{j+1} \\ 0, & \lambda > \lambda_{j+1} \end{cases}$$

where a is a proportion of approximation pieces and λ_j are the segment transition points. The value of a may be estimated empirically, but different types of curves will have a certain portion of overlap necessary for smooth transition. An example of a blended sample is given in the Figure 1.

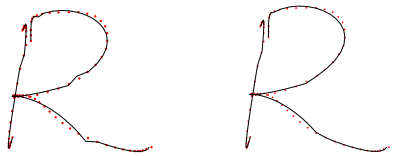


Figure 1. Example of blending.

5. Experiments

We have performed two sets of experiments using our compression method for both text and binary representations of curves. Before describing the experiments in detail, we describe the setting.

5.1 Experimental Setting

The dataset of handwritten samples was collected in the Ontario Research Center for Computer Algebra with various tablet devices. The highest resolution device's specifications were: 512 pressure levels, 2540 dpi resolution and 133 pps max data rate. The sampling error of the device was $\pm.02$ in and the resolution of the monitor was 94 dpi. Therefore, the absolute sampling error, as the stroke is rendered on the screen, is $\approx \pm 2$ pixels. Error, relative to the height of writing, is $\approx 2.5\%$.

Different individuals were asked to write various parts of regular text to ensure variations in the length of strokes and writing styles. Overall, we obtained 108,094 points split in 1,389 strokes.

Compressed size reported for the experiments is obtained by comparing the compressed size of the entire database to the original size, reporting it as a fraction between 0% and 100%.

5.2 Compression of Textual Traces

Representation One set of experiments used a textual representation of trace data. It may seem odd to explore methods to represent text more compactly as text, but this is relevant for standard XML representations.

For these tests we stored coefficients in UTF 8 format and define approximation packets as

$$\lambda_0; c^1_{00}, c^1_{01}, ..., c^1_{0d_{01}}; ...; c^N_{00}, c^N_{01}, ..., c^N_{0d_{0N}}$$
$$\lambda_1; c^1_{10}, c^1_{11}, ..., c^1_{1d_{11}}; ...; c^N_{10}, c^N_{11}, ..., c^N_{1d_{1N}} \quad ...$$
$$\lambda_D$$

where λ_i is the initial parameterization value of piece i in the stream, N is the number of channels (such as x and y coordinates of points, pressure, etc.) and d_{ij} is the degree of approximation of the piece i for j-th channel. Pen-based devices typically provide three channels: x, y coordinates of points and pen pressure p. In the example, the stream consists of D approximation pieces and the last packet defines the final value of parameterization. These packets define the approximation functions $f^j_i(\lambda)$, $i = 0..(D-1)$, $j = 0..N$ for corresponding intervals $[\lambda_i, \lambda_{i+1}]$. The end of a stroke can be defined with a special character, such as "%". The proposed model is independent of the choice of parameterization, which can be time, arc length, etc.

In the experiments below we found the combination of size of coefficients and degree that gives the best compression for error of 3%. Fixing these parameters, we estimate compression for other error values.

Size of Coefficients The next question was how dependent is the compression on the size of fractional part of coefficients. The result of the experiment for Chebyshev polynomials for the *fixed degree* method for different fractional sizes is presented in the Table 2 for different degrees of approximation for the maximal error of 3%. Even though the fractional size of 0 seems to be a more optimal combination for the given error threshold, for smaller error bounds it becomes inefficient. Therefore, we took the combination of the size of fractional part to be 1 and degree to be 7 and found the compressed size for other error thresholds. The same procedure was performed for other bases methods and the results are shown in Figure 3(a).

A similar experiment for Chebyshev polynomials has been performed for the *fixed length* method. Results are shown in the Table 3. Intervals that cannot be approximated with degree ≤ 20 are recorded as "–".

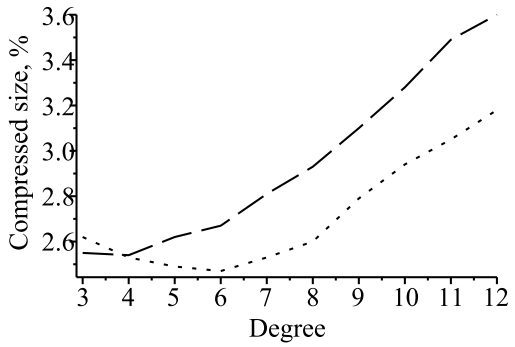


Figure 2. Compression for parameterization by time (dot) vs. arc length (dash) for series with integer coefficients. Compressed sizes measured for all (integer) degree values from 3 to 12.

We observed that the fixed degree method performs significantly better than the fixed length method, while eliminating the associated risk of intervals that can not be approximated. Therefore, all subsequent experiments were performed with variations of the fixed degree method.

Fixing the Size of Coefficients In the next experiment we asked whether compression would change if we took coefficients as real numbers with fixed number of digits. Similarly to the previous experiment, we looked at the rates for different degrees (1-12) and the number of digits to find the optimal combination for maximal error of 3%. Taking this combination, we then found compression for other values of maximal pointwise and RMS errors. We observed, however, that applying this algorithm literally is not a preferred solution, since the coefficient of the 0-th degree polynomial is usually significantly larger than the other coefficients. This can be explained by the fact that the degree 0 polynomial serves as a position translator. We therefore allow this coefficient to be twice of the size of coefficients of higher degree. In Table 4 results are shown for different degrees of approximation and the size of coefficients of degree > 0 for Chebyshev polynomials. Compression for other error threshold for all bases is given in Figure 3(b).

5.3 Compression of Binary Traces

We next explored how to compress data for applications that can store ink data in binary form. To do this we stored the sequence of approximation coefficients in an exponential format as ab where a and b are two's complement binary integers, standing for significand and a power of 10 respectively. We fixed the size of b to be 3 bits and changed only the size of a.

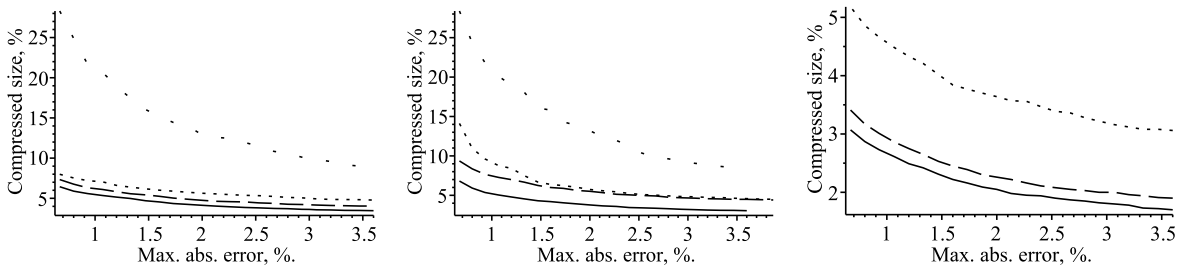


Figure 3. Compressed size for different values of error for Chebyshev (solid), Legendre (dash), Legendre-Sobolev (dot) and Fourier (spaced dot). Graph (a) is for coefficients with 1 digit fractional part. Graph (b) is for fixed coefficient size for the best degree (Chebyshev: F=2,D=7, Legendre: F=3,D=8, L-S: F=3,D=7, Fourier: F=2,D=6). Graph (c) is for adaptive binary representation with the best degree/coefficient size chosen dynamically from degrees between 1 and 12 and coefficients between 3 and 9 bits. Compressed sizes measured for maximum pointwise error from 0.5% to at least 3.6% in .1% increments.

F\D	3	5	7	9	11	13	15
0	2.62	2.49	2.53	2.79	3.05	3.31	3.59
1	3.91	3.69	3.62	3.70	3.69	3.70	3.64
2	5.36	5.18	5.10	5.29	5.24	5.27	5.21
3	6.82	6.65	6.58	6.87	6.81	6.84	6.80
4	8.29	8.13	8.07	8.45	8.37	8.42	8.38

Table 2. Compressed size (%) by degree of approximation (D) and fractional size of coefficients (F) for Chebyshev polynomials and max. error of 3%, fixed degree method.

F\L	10	20	30	40	50	60
0	4.67	–	–	–	–	–
1	6.79	5.01	4.29	4.09	3.87	–
2	9.10	6.81	5.94	5.74	5.45	–
3	11.22	8.63	7.59	7.37	7.04	–
4	13.43	10.44	9.23	9.01	8.63	–

Table 3. Compressed size (%) by length of intervals (L) and fractional size of coefficients (F) for Chebyshev polynomials and max. error of 3%, fixed length method.

S\D	3	5	7	9	11	13	15
2	2.95	2.99	3.00	3.15	3.20	3.19	3.35
3	4.39	4.44	4.48	4.71	4.71	4.76	4.76
4	5.85	5.92	5.96	6.30	6.26	6.32	6.32
5	7.31	7.39	7.46	7.88	7.82	7.90	7.92

Table 4. Compressed size (%) for different approximation degrees (D) and coefficient sizes (S) for Chebyshev polynomials with max. error of 3%, fixed degree method.

We note that the fixed degree and fixed length segmentation schemes have parameters whose optimal choice depend on the application. Certain types of strokes have their own optimal combination of parameters. This becomes especially noticeable when curve patterns have completely different styles: from straight line to curly handwriting. Therefore, for our final experiment, we used the adaptive segmentation scheme and chose stroke-wise approximation parameters for each input channel separately. Compression packets for each stroke i took the form

$$b_i; d_i; \lambda_1; c_{10}, ..., c_{1d_i}, \lambda_2; c_{20}, ..., c_{2d_i} \ \ ... \ \ \lambda_D$$

where b_i is the number of bits, d_i degree, λ_j initial value of parameterization of piece j and $c_{j0}, c_{j1}, ..., c_{jd_i}$ are coefficients. This method gives significantly better compression, as shown in Figure 3(c) and Table 5. Compression with Chebyshev polynomials for 1% maximum error yields 2.6% compressed size, for 2.5% (sampling error of the device) it yields 1.9% size. A maximum error of $< 2.5\%$ is indistinguishable by a human and such compression can be accepted as equivalent to lossless for the most of applications in pen-based computing.

5.4 Comparison with Second Differences

The second differences method yields high compression for low-resolution devices. Compression for high-resolution devices is lower, at the same sampling rate, because of higher variance. A stroke may be represented by the values of the first two points and a sequence of second differences, since $x_{i\triangle 2} = x_i - 2x_{i-1} + x_{i-2}$. We stored these values as binary numbers of fixed size, similar to the manner described in Section 5.3. The size for the first two values was different from the size of second differences. Because the method is lossless, it was necessary to use a number of coefficient bits sufficient to store all values exactly. We then performed gzip encoding on this binary stream to model the compression algorithm described in [12]. For the handwriting collected with our device (see Section 5), this method yields compression to 8.64%.

B\E,%	0.0	0.6	1.1	1.5	2.0	2.5	3.1	3.5
C	–	7.50	6.22	5.93	5.26	5.14	4.87	4.65
L	–	9.22	6.97	6.32	5.64	5.25	5.20	5.04
L-S	–	12.64	11.21	10.19	8.67	8.55	8.26	7.51
$\triangle 2$	23.35	–	–	–	–	–	–	–

(a) binary coefficients

B\E,%	0.0	0.6	1.1	1.5	2.0	2.5	3.1	3.5
C	–	3.07	2.61	2.31	2.05	1.90	1.80	1.72
L	–	3.41	2.86	2.53	2.26	2.08	2.00	1.91
L-S	–	9.36	7.27	6.25	5.51	4.98	4.64	4.49
$\triangle 2$	8.64	–	–	–	–	–	–	–

(b) binary coefficients, compressed

Table 5. Compressed size (%) for binary representation with different pointwise error limits (E) and bases (B): Chebyshev (C), Legendre (L) and Legendre-Sobolev (L-S). The lossless second difference method is shown for comparison ($\triangle 2$).

The approach of representing handwriting by orthogonal series approximation has an important advantage other than better compression. It allows to build the database of handwritten samples and use it almost directly in recognition algorithms [7] without recomputing the coefficients. It does not restrict classification method to a specific orthogonal basis, since the basis can be changed by one matrix multiplication (albeit possibly with high condition number).

6. Conclusion and Future Work

We have presented an approach to compression of digital strokes using functional approximation. We have shown that Chebyshev polynomials give very high compression and allow flexible approximation with desired accuracy. The compressed format of written samples serves as a suitable input for the character classification algorithms [7] and allows to integrate compression and recognition in a unified efficient infrastructure.

Certain other algorithms may prefer to use Legendre and Legendre-Sobolev bases as they allow online moment computation and function recovery [6]. One can gain the most space advantage by storing compressed strokes represented by Chebyshev coefficients and converting them to Legendre or Legendre-Sobolev format.

The relationship between the precision of coefficients in different bases is affected by the condition number of conversion matrix. For conversion from Chebyshev to Legendre basis in the range of degrees, d, of interest, the condition number is approximately $0.15 + 0.73d$. The condition number for conversion to Legendre-Sobolev basis is approximately $3.74^{d-0.40}$.

For future work, an interesting topic is to estimate the relationship between compression and recognition for different orthogonal bases. Another important aspect is to consider different error measures – computing the Legendre-Sobolev distance allows to estimate quality of approximation in the first jet space where other error measures may be more appropriate.

References

[1] I. Al-Jarwan and M. Zemerly. Image compression using adaptive variable degree variable segment length Chebyshev polynomials. *Springer LNCS*, 3540/2005:1196–1207, 2005.

[2] B. W. Char and S. M. Watt. Representing and characterizing handwritten mathematical symbols through succinct functional approximation. In *Proc. ICDAR*, pages 1198–1202. IEEE Computer Society, 2007.

[3] M. Chatterjee. System and method for ink or handwriting compression. *United States Patent No US 6,549,675 B2*, April 2003.

[4] J. Ferraiolo, F. Jun, and D. Jackson. *Scalable vector graphics (SVG) 1.1 specification*. W3C, January 2003.

[5] O. Golubitsky, V. Mazalov, and S. M. Watt. Orientation-independent recognition of handwritten characters with integral invariants. In *Proc. Joint Conf. ASCM 2009 and MACIS 2009*, volume 22 of *COE Lecture Notes*, pages 252–261, Japan, Dec. 2009. Kyushu University.

[6] O. Golubitsky and S. M. Watt. Online stroke modeling for handwriting recognition. In *Proc. CASCON '08*, pages 72–80, New York, NY, USA, 2008. ACM.

[7] O. Golubitsky and S. M. Watt. Distance-based classification of handwritten symbols. *International J. Document Analysis and Recognition*, 13(2):113–146, 2010.

[8] I. Guyon. *Unipen 1.0 Format Definition*. AT&T Bell Laboratories, 1994. `http://unipen.nici.ru.nl/unipen.def`.

[9] D. A. Huffman. A method for the construction of minimum-redundancy codes. In *Proceedings of the I.R.E.*, pages 1098–1102, September 1952.

[10] H. Levitt and A. Neuman. Evaluation of orthogonal polynomial compression. *Journal of the Acoustical Society of America*, 90(1):241–252, July 1991.

[11] Z. Liu, H. S. Malvar, and Z. Zhang. System and method for ink or handwriting compression. *United States Patent No US 7,302,106 B2*, November 2007.

[12] Microsoft Inc. *Ink serialized format specification*.

[13] Slate Corporation. *Jot - a specification for an ink storage and interchange format*, May 1996. `http://unipen.nici.kun.nl/jot.html`.

[14] S. M. Watt and T. Underhill (editors). *Ink markup language (InkML)*. W3C Working Draft, May 2010.

[15] H. Zenkouar and A. Nachit. Images compression using moments method of orthogonal polynomials. *Materials Science and Engineering B*, 49(3):211–215, 1997.

[16] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23:337–343, 1977.