Geometric Techniques for Digital Ink

(Spine Title: Geometric Techniques for Digital Ink)

(Thesis format: Monograph)

by

Vadim Mazalov

Graduate Program
in
Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO
THE SCHOOL OF GRADUATE AND POSTDOCTORAL STUDIES

CERTIFICATE OF EXAMINATION

Supervisor:	Examination committee:
Dr. Stephen M. Watt	Dr. Steven Beauchemin
	Dr. Mahmoud El-Sakka
	Dr. Greg Reid

The thesis by

Vadim Mazalov

entitled:

Geometric Techniques for Digital Ink

is accepted in partial fulfillment of the
requirements for the degree of
Master of Science

Date	Chair of the Thesis Examination Board

Abstract

Handwriting is one of the most natural ways for a human to record knowledge. In recent years this type of human-computer interaction has received increasing attention due to the rapid evolution of digital ink hardware. This thesis contributes to the art of efficient recognition of handwriting and compact storage of digital ink.

In the first part of the thesis, we focus on the development of algorithms for transformation-invariant recognition of handwritten mathematical characters. We first implement a rotation-independent classification method based on the theory of integral invariants of parametric curves. We then extend this method to shear-invariant recognition. Presence of affine transformations creates difficulties in parameterization of coordinate functions and size normalization of handwritten samples. We therefore present an affine-invariant size normalization approach and develop a mixed parameterization, which is insensitive to large affine transformations and yields a relatively high recognition rate.

In the second part of the thesis, we develop digital ink compression algorithms taking advantage of the theory of approximation of curves with orthogonal polynomial series. We then test the compression rate for Chebyshev, Legendre and Legendre-Sobolev orthogonal polynomials, as well as for Fourier series. By studying the compression ratio for representing coefficients in Unicode and binary formats, we show that Chebyshev polynomials give the best compression rate and can be successfully used in related applications.

Keywords: Recognition of handwritten characters; Rotation-invariant recognition; Shear-invariant recognition; Pen-based computing; Compression of digital ink

Acknowledgements

First of all, I am thankful to my supervisor, Prof. Stephen M. Watt. His dedication, guidance and brilliant ideas created the most enjoyable and fruitful atmosphere and served as the generator for many of the experiments presented in the thesis.

My special appreciation goes to Dr. Oleg Golubitsky for his participation and valuable suggestions. I also thank my friends – the faculty and student members of the Ontario Research Center for Computer Algebra – for help and encouragement.

Above all, this work would not be possible without my parents, my brother and sister. Words can not express my gratitude to you for promoting my creative growth.

Chapters of this thesis are based on co-authored papers accepted to certain conferences. Chapter 4 is based on a paper accepted to the Joint Conference of ASCM 2009 and MACIS 2009: Asian Symposium of Computer Mathematics and Mathematical Aspects of Computer and Information Sciences. Chapter 5 is based on a paper accepted to the 2010 International Workshop on Document Analysis Systems. Chapter 6 is based on a paper accepted to the 12th International Conference on Frontiers in Handwriting Recognition, ICFHR 2010. I thank again the co-authors, Stephen M. Watt and Oleg Golubitsky, for the productive collaboration in these areas.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the well-established popularity of hand-held mobile and digital tablet devices, online recognition of handwritten mathematics has received increasing attention in recent years. This subarea of handwriting recognition allows two-dimensional input of mathematical expressions in a more natural way than other alternatives. This is preferable in some settings, in which a keyboard is not accessible or is intentionally avoided, e.g. during a lecture or online scientific collaboration [34].

Although considerable work has been done in the field of handwriting recognition, the classification of mathematical symbols requires special attention. Among the factors that give classification of mathematics additional challenges beyond those of normal text recognition, is the relatively large "alphabet" of similar looking few-stroke symbols that can be subjected to transformations, such as scale, rotation and shear. The absence of a fixed dictionary of multi-symbol "words" creates limitations for syntactic verification of recognized formulas. The two-dimensional nature of mathematical expressions requires an accurate differentiation between fluctuations in positioning and intentional super- or sub-scripting over a baseline. In this context, character classification algorithms for handwritten mathematics require special consideration.

Online recognition deals with assigning a sample, given by a vector of coordinates, to a particular class based on a dataset of training samples. It is important to note that online classification, as opposed to offline pattern matching, aims to minimize the amount of computation after pen-up, even if it increases computations after a sample is being written.

In an online classification environment, a curve is given as an ordered set of points in a Euclidean plane. Tablet devices are capable of capturing coordinates of a stylus as functions of time. Therefore, the input to an online classification algorithm is typically given as a vector of pen coordinates represented as real numbers and spaced on a fixed time interval. In addition, some devices can collect other information, such as the degree of pressure or pen angle, as well as coordinates of pen-up points, i.e. when a stylus does not touch the screen. We however do not consider this information to maintain independence of certain hardware features and devices.

A curve is given as a sequence of tuples

$$(x_0, y_0, t_0), (x_1, y_1, t_1), ..., (x_n, y_n, t_n)$$

where $x_i, y_i, t_i \in \mathbb{R}, i = 0..n$, and $t_0 < t_1 < ... < t_n$.

Points are usually considered equally spaced in time and therefore t_i can be omitted. Therefore, a sample character can be represented as shown in Figure 1.1.

Stroke 1: [(7515, -7870), (7516, -7877), (7515, -7900), (7517, -7937), (7520, -7964), (7518, -7989), (7520, -8022), (7520, -8054), (7522, -8088), (7522, -8119), (7525, -8144), (7526, -8163), (7532, -8188), (7541, -8196), (7555, -8199), (7589, -8184), (7612, -8177), (7640, -8167), (7668, -8157), (7697, -8137), (7730, -8130), (7753, -8120), (7780, -8118), (7803, -8115), (7819, -8113)]

Stroke 2: [(7754, -7870), (7748, -7900), (7742, -7938), (7737, -7982), (7739, -8040), (7740, -8096), (7747, -8152), (7761, -8210), (7774, -8263), (7784, -8305), (7795, -8344)]

Coordinates are most often given as integers, and indeed that is how the dataset that we use in our experiments is stored.

The rest of the thesis is organized as follows. In Chapter 2 we discuss some of

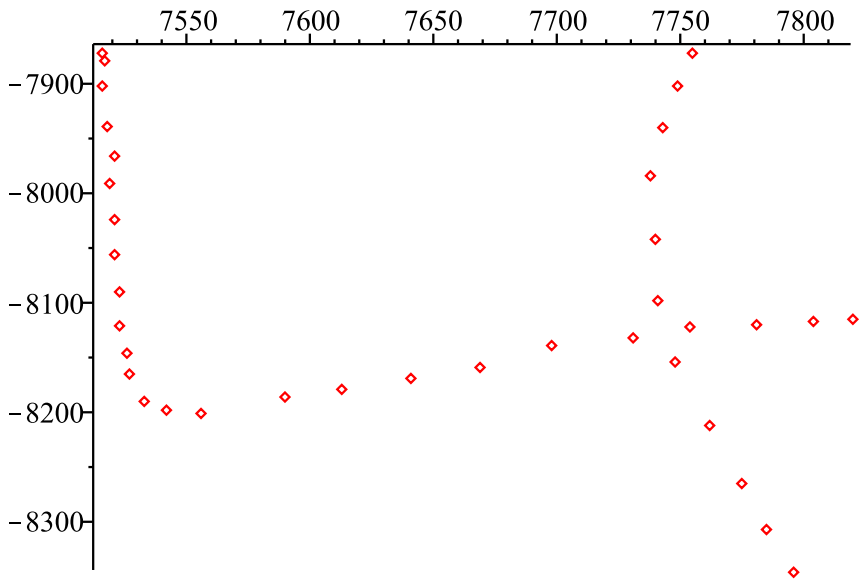


Figure 1.1: An example of a 2-stroke sample

the previous work and preliminaries. We look at the method called "elastic matching" and at the theory of approximation of curves with orthogonal polynomials. We explain the method of recognition based on support vector machines (SVMs) and distance to convex hulls of nearest neighbours. We present a possible extension of this theory to recognition of multi-stroke characters. In the same chapter we show that the method developed by Golubitsky and Watt [14] is not capable of recognizing characters subjected to distortions. We then study some existing techniques for classification of transformed samples: geometric moments for rotated samples; and several methods for affine-invariant recognition: stroke-based affine transformations, minimax classification with hidden Markov models (HMM) and affine moment invariants. The basic ideas of our methodology and experimental settings are presented in the end of the Chapter.

In Chapter 3 we outline the major concepts of the theory of integral invariants of parametric curves and measure the quality of approximation of invariants.

In Chapter 4 we study rotation invariant recognition. We first show how to com-

pute coefficients of coordinate functions and integral invariants in a time-efficient manner. We then develop algorithms for rotation invariant recognition based on the coefficients of approximation of integral invariants with Legendre-Sobolev orthogonal polynomials. We show that performance of integral invariants is significantly higher than that of geometric moment invariants.

In Chapter 5 we develop a recognition approach, invariant with respect to shear transformations. We study some of the size normalization and coordinate functions parameterization methods. We propose a new method to normalize handwritten characters when affine distortions take place. Moreover, we propose a mixed parameterization that allows us to achieve a relative invariance to affine transformations while keeping the recognition rate quite high. Evaluation of the results is given at the end of the Chapter.

In Chapter 6 we develop a method for compact storage of online handwriting. We test variations of concepts of curve parameterization, segmentation and segment blending. We develop a binary packets format for storing a trace. We then compare our results with one of the most popular methods available today – compression with the second differences. The results show that our algorithm performs significantly better.

Chapter 7 concludes this thesis.

Chapter 2

Previous Work and Preliminaries

This chapter presents background concepts used in this thesis. Some, such as the theory of orthogonal series, are used throughout. Others, such as elastic matching, are discussed for comparison.

2.1 Elastic Matching

Elastic matching is derived from dynamic programming algorithms initially used in string matching problems. In such a setting, a string was represented as a vector. The membership of the vector in one of given classes was evaluated based on the squared Euclidean distance from the vector to corresponding training samples. In vision, elastic matching has been actively used in handwriting recognition, as well as in more sophisticated algorithms for image analysis [2, 39].

The elastic matching algorithm aims to minimize the total distance $D(n, m; k)$ between a test sample of n points and the training sample k of m points, as shown

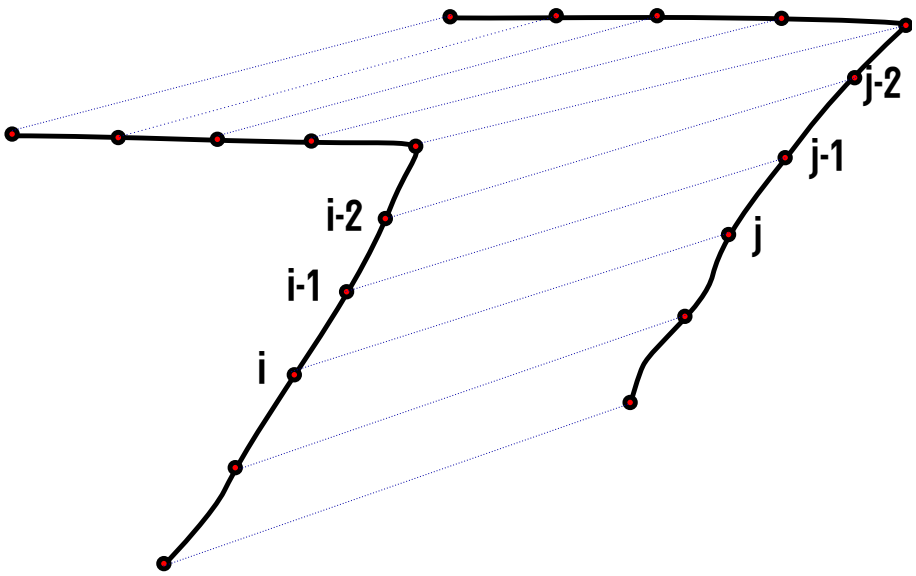


Figure 2.1: Elastic Matching

in Figure 2.1

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

where $d(i,j;k)$ is the square Euclidean distance between points i and j (see [32] for details). Several authors proposed to include additional attributes in the distance formula to capture certain stroke features, i.e. curve orientation, defined as the difference between angles of slopes. Therefore, the distance formula between points i and j may be written as [32]

$$d(i,j) = (x_i - x_j)^2 + (y_i - y_j)^2 + \mu|\alpha_i - \alpha_j|$$

where (x_i, y_i) is i-th point of the test sample, (x_j, y_j) is j-th point of the training

sample, μ is a constant that can be found empirically and

$$\alpha_i = \arccos\left(\frac{x_{i+1} - x_i}{\sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}}\right).$$

Finding the minimum distance between two patterns is expensive, especially if a character consists of hundreds of points. There have therefore been several attempts to make the algorithm more efficient, such as iterative elastic matching [32]. However, it still requires relatively many arithmetic operations when compared, for example, to classification of curves approximated with truncated orthogonal series. The situation becomes even worse in the case of handwritten mathematics because of the large number of classes. As a result, this methodology may not be suitable for a deployment on mobile devices with decent computational capabilities.

2.2 Approximation of Curves with Orthogonal Series

To overcome the disadvantages of elastic matching, Char and Watt proposed to represent a character as a vector of coefficients of the approximation of the curve coordinates with truncated orthogonal series [4]. This approach helped to decrease the number of computations to the dimension of coefficient vectors, which is typically less than 25. We summarise the basic ideas here.

Two functions $f(t)$ and $g(t)$ defined on the domain $[a, b]$ are said to be orthogonal on this interval with respect to a given continuous weight function $w(t)$, if their inner product

$$\langle f, g \rangle \equiv \int_a^b f(t)g(t)w(t)dt = 0.$$

A well-known technique of approximation of a function $f : R \to R$ is finding a linear

combination of functions from truncated basis $P = \{P_i : R \to R, i = 0, 1, ..., d\}$:

$$f(t) \approx \sum_{i=0}^{d} c_i P_i(t), \quad c_i \in R,\ P_i \in P$$

where polynomials $P_i, i = 0, 1, ..., d$ are orthogonal with respect to an inner product $\langle \cdot, \cdot \rangle$. The system of orthogonal polynomials $\{P_0, P_1, ...\}$ with respect to a given inner product can be obtained with Gram-Schmidt orthogonalization on the monomial basis $\{1, t, t^2, ...\}$. The coefficients c_i can be computed as

$$c_i = \frac{\langle f, P_i \rangle}{\langle P_i, P_i \rangle}.$$

Following this technique, one is able to obtain the following representation of coordinate functions:

$$X(t) \approx \sum_{i=0}^{d} x_i P_i(t), \qquad Y(t) \approx \sum_{i=0}^{d} y_i P_i(t).$$

Note that $X(t)$ and $Y(t)$ can be parameterized by time, arc length, or other parameterization choices. Parameterization by arc length is preferable over parameterization by time, since it provides independence of variations in speed of writing. Depending on the setting, a potential problem is that parameterization by arc length is not invariant under affine transformations. There is, however, parameterization by special affine arc length, invariant under area preserving transformations [1]:

$$F(L) = \int_0^L \sqrt[3]{x'(t)y''(t) - x''(t)y'(t)}dt.$$

It was proposed in [4] to approximate coordinate functions with Chebyshev polynomials of the first kind

$$T_n(t) = \cos(n \arccos t).$$

These are orthogonal on the interval $[-1, 1]$ for $w(t) = 1/\sqrt{1 - t^2}$. While Chebyshev

polynomials are easy to compute and allow accurate approximation of a curve with low degree series, the form of its weight function creates difficulties for online computation of approximation. Therefore Golubitsky and Watt [10] proposed to use Legendre polynomials that perfectly fit into the model of recovering a function online from its moments [35]. They showed how to compute the first d coefficients of the truncated Legendre series for some function $f(\lambda)$, normalized to a desired range and domain, in online time $OL_n[O(d), O(d^2)]$, where n is the number of known equally-spaced values of f.

In later work, the authors showed that Legendre polynomials are outperformed by Legendre-Sobolev polynomials. The latter polynomials have the inner product of the form

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

Legendre-Sobolev polynomials still allow online computation of coefficients, while providing a more accurate description of a curve for a lesser degree of approximation (due to the presence of derivatives in the formula for inner product) [11]. Recognition is based on Euclidean distance measure between coefficient vectors of subject and training samples. In the same work, the authors showed that classification rates with elastic matching and Legendre-Sobolev approximation are similar, while the latter is more efficient.

2.3 Classification with SVMs

Once a curve is represented with coefficients of its approximation, it can be recognized with the support vector machine (SVM) classifier. SVMs have been successfully used in various areas of pattern recognition. The aim of SVM classification is to find the separating boundary of two given classes that gives the maximal distance between

the boundary and each class. Such a boundary function can be expressed as [20]

$$f(x) = \sum_i (a_i y_i K(x, x_i)) + b$$

where x_i represents subject sample, y_i stands for a training sample and $K(x, x_i)$ is the kernel function. Parameters a_i and b are obtained by minimization of the function with certain constraints [36].

2.4 Classification with Convex Hulls

This section is based on the paper "Orientation-independent recognition of handwritten characters with integral invariants" [28] co-authored with Golubitsky and Watt.

The technique of recognition via convex hulls represents classes by some fixed number of nearest neighbours and is similar to the recognition with SVMs. However, a subject sample is assigned to the class with a corresponding convex hull located on the smallest distance to the sample. Nearest neighbours are selected with the Manhattan distance, which is among the fastest distances known, requiring $2d - 1$ arithmetic operations, where d is the dimension. Distance to convex hulls is evaluated with the squared Euclidean distance, which takes $3d - 1$ operations.

Computing the distance from a point to a convex hull is generally expensive. However, one can represent a convex hull as a simplex if the number of nearest neighbours is less than the dimension of the vector space and the points are in generic position. If the points happen to not be in generic position, a slight perturbation is done with a little affect on the distance. The algorithm is then recursively iterated until the projection of the point on the smallest affine subspace containing the simplex happens to be inside the simplex. This algorithm has complexity of $O(N^4)$, where N is the dimension of the vector space. However, since at each recursive call the dimension often drops by more than one, in practice this algorithm is less expensive [13].

2.5 Classification of Multi-Stroke Characters

It was shown in [12] that classification of multi-stroke characters can be implemented similarly to the classification of a single-stroke with functional approximation. In the case of a multi-stroke sample, consecutive strokes are joined to obtain the function to approximate, and the number of strokes is included in the class label.

2.6 Previous Work and Dependence on Distortions

This section is based on the paper "Toward Affine Recognition of Handwritten Mathematical Characters" [29] co-authored with Golubitsky and Watt.

Some work has been done [4, 10, 14, 11] in recognition based on the approximation of coordinate functions by truncated polynomial series. Different bases have been studied, including Chebyshev, Legendre and Legendre-Sobolev bases. Legendre-Sobolev series were chosen as the most suitable, since these polynomials are easy to compute and provide a constructive distance measure in the first jet space, taking derivatives into account. Subject samples are assigned to training classes with distance measure from the sample to convex hulls in the space of Legendre-Sobolev coefficients. Recognition rate of this method is 97.5% for a dataset of individual symbols in our collection. Although the samples do exhibit a certain amount of affine distortions, it is expected that characters written in real-life, as a part of a mathematical formula, are likely to be much more affected by such transformations. Indeed, most users of pen-based devices tend to write symbols with a certain degree of rotation and/or shear. These transformations create difficulties for classification of samples. To measure the dependence of the algorithm developed in [4, 10, 14, 11] to distortions, characters were randomly rotated on some interval. In the experiment we

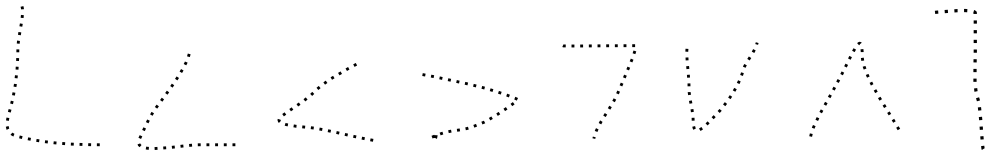


Figure 2.2: Ambiguity introduced by shear and rotation

observed that the classification rate for the algorithm has approximately quadratic dependence on the angle and decreases to 90% when the rotation angle ranges in the interval of $[-0.3, 0.3]$ radians. We conclude that this algorithm is quite sensitive to transformations and remains no longer robust when affine distortions take place [29].

Rotation invariant classification is a more challenging task compared with the problem of size and position normalization. However, it is not as advanced as shear and, more generally, affine invariant recognition. The main difficulty is that different characters may require different correction and the type and degree of such correction is not known in advance. With mathematical handwriting, it is especially challenging to detect the dominant distortion from symbol features, even when characters are analyzed in a set. Another issue, specific to mathematical symbols, is the requirement of careful analysis of transformation limits to avoid blending classes. For instance, when a sample L is the subject to shear transformation, it can be easily misclassified with $\angle$. If, in addition, we allow arbitrary rotation and scaling of the character, the set of matching candidates will include $<$, $>$, 7, V, $\bigwedge$, $\rceil$, $\surd$, Γ and $\wedge$, as shown in Figure 2.2 [28].

Another challenge with shear and affine transformations is the size normalization, since the regular techniques are no longer suitable. Curve parameterization becomes not as trivial either, since the arc length – the most common parameterization choice – is not invariant under affine transformations.

2.7 Rotation Invariance with Geometric Moments

This section is based on the paper "Orientation-independent recognition of handwritten characters with integral invariants" [28] co-authored with Golubitsky and Watt.

Moment invariants are a widely used toolset to describe curves independently of orientation. Among moment functions one can select geometric, Zernike, radial and Legendre moments [30]. For the purpose of online character recognition under pressure of computational constraints, geometric moments appear to be the most attractive, since they are easy to calculate and provide invariance under scaling, translation and rotation [16]. Geometric moments have been widely used in pattern classification [7, 23, 30]. A $(p+q)$-th order moment of a function $f(x,y)$ can be expressed as

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y).$$

As it was originally defined, translation invariance is achieved by computing central moments

$$\mu_{pq} = \sum_x \sum_y (x-x_0)^p (y-y_0)^q f(x,y),\ x_0 = \frac{m_{10}}{m_{00}} \text{ and } y_0 = \frac{m_{01}}{m_{00}}$$

while scale normalization is performed as

$$\eta_{pq} = \mu_{pq} / (\mu_{00})^{(p+q+2)/2}$$

There is an infinite family of moment invariants, derived from algebraic invariants, and the first three can be represented as

$$M_1 = \eta_{20} + \eta_{02},$$
$$M_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2,$$
$$M_3 = \eta_{20}\eta_{02} - \eta_{11}^2.$$

The independence of orientation of the above expressions can be verified by substitution with the geometric moments obtained after rotational transformation

$$
\begin{aligned}
m'_{20} &= \frac{1+\cos 2\alpha}{2} m_{20} - \sin 2\alpha \; m_{11} + \frac{1-\cos 2\alpha}{2} m_{02}, \\
m'_{11} &= \frac{\sin 2\alpha}{2} \; m_{20} + \cos 2\alpha \; m_{11} - \frac{\sin 2\alpha}{2} \; m_{02}, \\
m'_{02} &= \frac{1-\cos 2\alpha}{2} m_{20} + \sin 2\alpha \; m_{11} + \frac{1+\cos 2\alpha}{2} m_{02}.
\end{aligned}
$$

One can omit translation and scale normalization of moments by, first, normalizing a sample's coordinates. In this case, the moment invariants are derived in terms of moments m_{pq}.

2.8 Affine-Invariant Recognition

This section explores existing approaches to affine-invariant recognition and is based on the paper "Toward Affine Recognition of Handwritten Mathematical Characters" [29] co-authored with Golubitsky and Watt.

2.8.1 Stroke-Based Affine Transformation

If a sample is subjected to an arbitrary affine transformation, one could estimate the transformation matrix by solving a minimization problem. This idea was implemented in [37], where it was proposed that stroke-based affine transformations should be applied to a character in order to find the distortion of a test sample that gives the minimal distance to each training class. The algorithm denotes stroke-wise uniform affine transformation for a stroke i with A_i and b_i, where A_i is a 2×2 matrix for shear, rotation and scale and b_i is a 2-dimensional translation vector. The objective function for the minimization problem for a character of N strokes is constructed in

the form of least-squares data fitting to determine the optimal A_i and b_i

$$F_i = \sum_k \|A_i t_k + b_i - r_k\|^2 \rightarrow \min \text{ for } A_i, b_i, (1 \leq i \leq N)$$

where t_k and r_k are the k-th feature points of the sample to be classified and a reference sample respectively, and $\|\cdot\|$ is the Euclidean norm. The solution to this problem gives the affine transformation that yields the least distance between corresponding strokes of the test and a training sample. This procedure is performed for each training sample before distance based classification takes place.

This idea was extended to recognition of handwritten characters as grayscale images [38]. Due to the computationally intensive nature of the algorithm, it may be applicable for such offline pattern classification.

2.8.2 Minimax Classification with HMM

An online method, robust to affine distortions, was developed in [18] and is based on continuous-density hidden Markov models (CDHMM). Let N be the number of character classes $C_i,\ i = 1, ..., N$, each containing M_i CDHMMs

$$\left\{\lambda_i^{(m)}, m = 1, ..., M_i\right\}.$$

In the non-affine method, input symbol I is classified as a member of class C_i by representing it as

$$i = \arg\max_j \left\{\max_m \left[\max_S \log p(I, S|\lambda_j^{(m)})\right]\right\}.$$

where $p(I, S|\lambda_j^{(m)})$ denotes the joint likelihood of the observation O and the associated hidden state sequence S with given CDHMM $\lambda_j^{(m)}$. To eliminate affine distortions

between the input and training samples, it is proposed to evaluate

$$i = \arg\max_j \left\{ \max_m \left[\max_S \log p(I, S|\Gamma_{\hat{A}}(\lambda_j^{(m)})) \right] \right\},$$

where Γ_A is a specific transformation of $\lambda_j^{(m)}$ with parameters A, and

$$\hat{A} = \arg\max_A p(I, \Gamma_A(\lambda_j^{(m)})).$$

The authors solve this problem with three iterations of the EM algorithm described in [21].

2.8.3 Affine Moment Invariants

Geometric moment invariants are a useful tool for rotation-independent recognition. An extension of this theory, proposed in [9], is called affine moment invariants (AMIs). AMIs are defined in terms of moments and independent of actions of the general affine group. Therefore, they have been successfully applied in recognition of handwritten samples. A central moment of order $p + q$ for a 2-dimensional object O is defined as

$$\mu_{pq} = \iint_O (x - x_c)^p (y - y_c)^q dx dy$$

where (x_c, y_c) is the center of gravity of the considered object O. In [9] an affine invariant description of a symbol is obtained in the form of a 4-dimensional vector of real numbers. This vector is composed of the first four AMIs. Classification is based on computing the Euclidean distance to classes with training symbols. Performance of AMIs is compared to the performance of regular geometric moment invariants for distorted handwritten samples, even though the latter is invariant under rotation, scale and translation. The results provided in this paper support the fact that AMIs

yield a better recognition rate than geometric moments for samples subjected to affine distortions.

2.8.4 Our Methodology

As opposed to the methods of stroke-based affine transformation and minimax classification with HMM, we propose a technique of classifying handwritten characters with integral invariants. Our approach is similar to the classification with AMIs – we calculate affine-invariant quantities from the original sample, without any a priori transformations of the curve. The difference between AMIs and integral invariants is that the former, as originally defined, provide curve-to-value correspondence, unlike curve-to-curve correspondence with the integral invariants. That means that an AMI is a value, while an integral invariant is a function. Integral invariants allow one to obtain a richer description of coordinate functions without excessive computations. In fact, we deploy only two invariants and find them sufficient for an acceptable classification accuracy of characters under different transforms. To increase the recognition rate even further, we still perform an analysis of a sample to obtain a numerical measure of the distortion (e.g. the angle of rotation or shear). However, we perform this analysis on a minor subset of classes, closest to the test sample in the space of coefficients of approximation of integral invariants. Such analysis is not computationally intensive.

2.9 Compression of Digital Ink

This section is based on the paper "Digital Ink Compression via Functional Approximation" accepted to the 12th International Conference on Frontiers in Handwriting Recognition, (ICFHR 2010), co-authored with Watt [25].

2.9.1 Ink Representation

A variety of digital ink standards are in use today. Among others, it is worth mentioning the vendor-specific or special-purpose formats: Jot [33], Unipen [15], Ink Serialized Format (ISF) [26] or Scalable Vector Graphics (SVG) [8]. In 2003, W3C introduced a first public draft of an XML-based markup language for digital trace description, InkML. This evolved into the current standard definition in 2006 [6]. InkML has received an increasing attention due to its vendor neutrality and XML based properties: simplicity, extendability, compatibility with other technologies and standards, platform-independence, portability, and wide support from software vendors. In addition, InkML, as an up-to-date standard, reflects most of the features provided by modern pen-based devices, e.g. pen tip pressure, pen tilt, etc.

An example of a trace, encoded in InkML, is given on the Listing 2.1. A digital trace is given as a sequence of points in the form (x, y, p), where (x, y) are coordinates of the point and p may stand for the pen pressure. In the general case, a trace is given in InkML as a sequence of n-dimensional entries $(v_1, v_2, ..., v_n)$. Each coordinate gives the value of a particular channel at that point. Assuming a character consists of 50 points and each value t_i requires 2 bytes of storage, it would require $\approx 100n$ bytes to store the set of points representing a symbol. Interchange and storage of such arrays becomes cumbersome, particularly when compared, for example, with 2 byte representation of a 16-bit Unicode character. It follows that techniques for ink data compression are worth detailed investigation.

2.9.2 Other Ink Compression Methods

One of the earliest ink compression methods was proposed in the JOT standard in 1993 and was based on the approximation with Bezier curves. The last available JOT standard [33] has a brief description of "compacted point format" that allows a storing application to skip a number of points and a reading application is supposed to insert

```
<traceFormat>
    <channel name="X" type="integer"/>
    <channel name="Y" type="integer"/>
    <channel name="F" type="integer"/>
</traceFormat>
<trace>
    3145 12112 176, 3147 12116 178, 3149 12119 175,
    3153 12124 174
</trace>
```

Listing 2.1: Example a trace in InkML format

the missing coordinates interpolating existing points. Since the interpolated values are obtained from the limited subset of original points, a noticeable approximation error is unavoidable.

A lossy algorithm was presented in [22], based on stroke simplification. It suggests elimination of excessive points to form 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 pre-defined threshold. Intuitively, such simplification may lead to jaggy curves. The authors address this issue and propose to interpolate strokes on the decompression stage with Hermite splines. They also address the question of avoiding smoothing cusp segments of the original curve, which are identified by comparing the angle between adjacent points to the prescribed value. This method has disadvantages similar to those of the algorithm described above. Moreover since the interpolated values are obtained from a subset of original points, noticeable approximation error is unavoidable.

A lossless compression scheme was proposed in [26] and similarly in [5], in which the authors assume that difference between consecutive points varies by a factor not greater than three. The algorithm computes the second order differences of data items in each data channel. On the example of X coordinates, first order difference of X is computed as $X_{i\triangle 1} = X_{i+1} - X_i$ and is expected to change very little. Similar com-

putations are performed for Y coordinates. The next assumption is that the second order difference is even more correlated. Therefore, computing $X_{i\triangle 2} = X_{i+1\triangle 1} - X_{i\triangle 1}$ gives a sequence of values with lower variance. The second differences are well-suited for compression with an entropy encoding algorithm, and the authors propose to use the Huffman encoder. As a result, the compressed data occupies approximately six times less memory than the original points. This result is indeed promising (especially considering the lossless nature of the compression). Yet, we discovered that the difference between consecutive coordinates in our dataset of samples 2.10 ranges between 0 and several hundreds. In fact, there are some samples in the LaViola [19] dataset with the difference of more than 500. Therefore, we expect the second differences algorithm, in general, to have a more decent compression rate than reported in the patent [5].

Another method, proposed in [5], is called "substantially lossless" which allows compression error's magnitude not to exceed the sampling error magnitude. In this approach the original curve is split in segments and each segment is represented by some predefined shape, such as a polygon, ellipse, rectangle or Bezier curve. It is not, however, mentioned how the shapes are obtained from the curve or the compression rate produced by this approach.

A preferable alternative is piecewise functional approximation of a curve. Our approach allows flexible approximation with a desired level of precision. This is achieved either by increasing the degree of approximation or decreasing the segment arc length. In addition, this approach yields high compression, as shown in the experimental part of this section.

2.9.3 Orthogonal Bases

Polynomial bases have been extensively used in previous work. For details see, *e.g.*, [14] and works cited there. Here we summarize a few basic facts relevant to stroke compression.

Since the interval of orthogonality in the definition of most of the classical orthogonal series is $\tau \in [-1, 1]$, for the purpose of compression 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

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

Such approximation is performed for $X(\lambda)$ and $Y(\lambda)$ coordinate function of the curve.

2.9.4 Bases for Approximation

We wish to determine which bases will be useful for compression. We have investigated the following.

Chebyshev polynomials of the first kind, defined as $T_n(\lambda) = \cos(n \arccos \lambda)$, have the weight function $w(\lambda) = \frac{1}{\sqrt{1-\lambda^2}}$. Chebyshev polynomials are widely used as the basis for functional approximation. In [4] it was shown that Chebyshev polynomials are suitable for succinct approximation of character strokes and perform better than

Bernstein polynomials. Therefore, we consider implementing a stroke compression algorithm with this kind of orthogonal polynomial series.

Legendre Polynomials are defined as

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

and have the 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 [14].

A property of Legendre and Legendre-Sobolev orthogonal bases, as applied to online stroke modeling, is the ability to recover a curve from its moments. Moments may be computed in real time, while the stroke is being written, and the coefficients of the stroke are calculated on pen-up in constant time dependent only on the degree of approximation [10].

The Fourier Series on $[-L, L]$ is provided for comparison, since we are not restricted in our selection of approximation basis. Consider the function defined on $[a, b]$. We can approximate this function on the interval $[-L, L]$ with Fourier series for periodic functions

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

where

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

2.10 Experimental Setting

This section is based on the paper "Toward Affine Recognition of Handwritten Mathematical Characters" [29] co-authored with Golubitsky and Watt.

Our dataset of handwritten mathematical characters currently comprises 50,703 samples from 242 classes. These samples have been collected from several sources: 26,139 characters were gathered at the Ontario Research Center for Computer Algebra (special mathematical characters, Latin letters and digits), 9,762 samples (digits, Latin letters and mathematical symbols) from the LaViola database [19], and 14,802 samples (mostly digits) from UNIPEN handwriting database [15].

All the samples are stored in a single file in InkML format. The number of strokes is included in the class labels. Thus, if a character, such as "7" is written with different number of strokes, it will be placed in different classes, even if the shape of the character is identical. Although this raises the total number of classes to 378, we have found it to give better recognition rates compared to when the number of strokes is included in the feature vector [12].

To avoid confusion, all gathered characters had been visually inspected to discard symbols unrecognizable by a human. Symbols that look ambiguous to a human reader (those that may belong to more than one class) were labelled with all the corresponding classes. Classes that appear indistinguishable without context analysis were merged, such as x and $\times$; o, 0 and O. If there was at least one sample in the class that could be recognized by a human with confidence, we retained the label of the class. As a result, we obtained 38,493 samples assigned to single classes, 10,224 to 2 classes, 1,954 to 3 classes, 19 to 4 classes, and 13 samples to 5 classes. Additional

details of the experimental setting are given in [14]. To increase the precision of the approximation of integral invariants, we precomputed some terms in the formulas for I_1 and I_2 in Maple [24] using rational arithmetic.

All tests are implemented in the 10-fold cross-validation setting. To conduct this process, symbols were split randomly in 10 parts, preserving the proportional sizes of the sets. The normalized Legendre-Sobolev coefficients of coordinate functions, integral invariants and moment invariants were precomputed for all symbols and stored in separate files. We attempted to increase precision of computations by precomputing some coefficients in Maple [24], mostly using rational arithmetic [14].

Part I

Recognition of Handwritten Characters

Chapter 3

Integral Invariants in Character Recognition

Integral invariants is an elegant theory for planar and spatial curves description under affine distortions.

3.1 Integral Invariants

This section is based on the paper "Orientation-independent recognition of handwritten characters with integral invariants" [28] co-authored with Golubitsky and Watt. In terms of handwriting recognition, a symbol is given as a parameterized piecewise continuous curve defined by a discrete sequence of points (for details, see Chapter 1). For a symbol we compute certain integral quantities from the coordinate functions, which are then also functions of the curve parameterization. Exposing the sample to transformations results in the same invariant functions. As opposed to differential invariants, such integral invariants are relatively insensitive to small perturbations, and are therefore applicable to classification of handwritten characters with sampling noise.

As the name suggests, integral invariants are given in terms of integration. Out

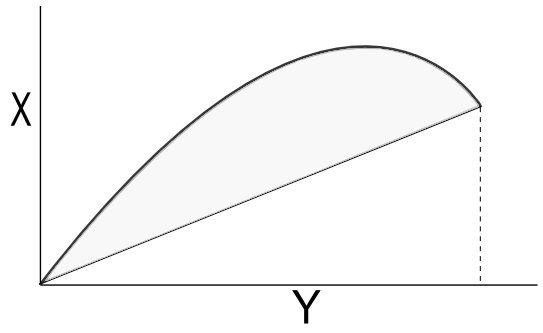


Figure 3.1: Geometric representation of the first order integral invariant

of the infinite family of invariants we study the first three [31], which we define in terms of the coordinate functions $X(\lambda)$ and $Y(\lambda)$:

$$I_0(\lambda) = \sqrt{X^2(\lambda) + Y^2(\lambda)} = R(\lambda),$$
$$I_1(\lambda) = \int_0^\lambda X(\tau)dY(\tau) - \frac{1}{2}X(\lambda)Y(\lambda),$$
$$I_2(\lambda) = X(\lambda)\int_0^\lambda X(\tau)Y(\tau)dY(\tau) - \frac{1}{2}Y(\lambda)\int_0^\lambda X^2(\tau)dY(\tau) - \frac{1}{6}X^2(\lambda)Y^2(\lambda).$$

Functions $X(\lambda)$, $Y(\lambda)$ can be of any desired parameterization. The function $I_1(\lambda)$ can be geometrically represented as the area between the curve and its secant (Figure 3.1).

Function $I_0(\lambda)$ is independent of transformations of the special orthogonal group $SO(2)$, while $I_1(\lambda)$ and $I_2(\lambda)$ are invariant under the group of special linear transformations, $SL(2)$. Full affine invariant can be obtained as the quotient

$$\frac{I_i(\lambda)}{I_1^i(\lambda)}\ , i = 2, 3, ...$$

We find it, however, to be less stable to compute than I_0, I_1 and I_2. Without loss of generality, by translating the origin and normalizing the size of a sample we focus our attention on the $SL(2)$ invariants.

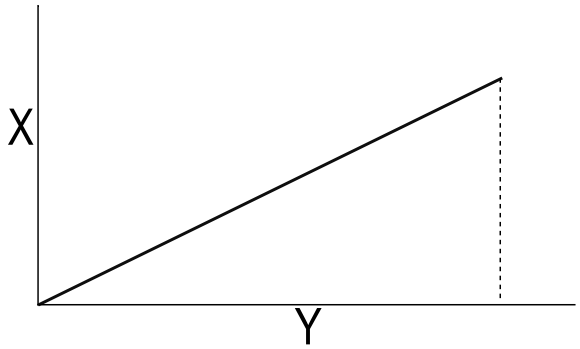


Figure 3.2: I_1 of a linear symbol

3.2 Approximation of Invariants

This section is based on the paper "Toward Affine Recognition of Handwritten Mathematical Characters" [29] co-authored with Golubitsky and Watt. Coordinate functions are represented by the truncated sum of Legendre-Sobolev orthogonal series (for details, see Section 2.2). Therefore, we can write the approximation of invariants introduced in the previous section as

$$
\begin{aligned}
I_0(\lambda) &\approx \sqrt{\left(\sum_{i=1}^{d} \bar{x}_i P_i(\lambda)\right)^2 + \left(\sum_{i=1}^{d} \bar{y}_i P_i(\lambda)\right)^2} \\
I_1(\lambda) &\approx \sum_{i,j=1}^{d} \bar{x}_i \bar{y}_j \left[\int_0^{\lambda} P_i(\tau) P_j'(\tau) d\tau - \frac{1}{2} P_i(\lambda) P_j(\lambda)\right] \\
I_2(\lambda) &\approx \sum_{i,j,k,l=1}^{d} x_i x_j y_k y_l \mu_{ijkl}
\end{aligned}
$$

where

$$
\begin{aligned}
\mu_{ijkl} = {} & P_i(\lambda) \int_0^{\lambda} P_j(\tau) P_k(\tau) P_l'(\tau) d\tau \\
& - \frac{1}{2} P_l(\lambda) \int_0^{\lambda} P_i(\tau) P_j(\tau) P_k'(\tau) d\tau - \frac{1}{6} P_i(\lambda) P_j(\lambda) P_k(\lambda) P_l(\lambda).
\end{aligned}
$$

Here P_i denotes the i-th Legendre-Sobolev polynomial.

In our algorithms, these functions are, in turn, approximated with the orthogonal

series. Therefore, it is reasonable to estimate how well the invariants can be approximated. To evaluate the quality of approximation, we compare coefficients of an original sample and the same sample sheared by 1 radian. Results are summarized in Table 3.1 in terms of the maximum error and the average relative error, defined as the quotient of the sum of absolute errors and the sum of absolute values, where degree is the degree of approximation.

Table 3.1: Maximum absolute and average relative errors in coefficients of invariants

	I_1		I_2	
Degree	Abs. Err.	Rel. Err.	Abs. Err.	Rel. Err.
2	9×10^{-12}	3×10^{-19}	3×10^{-11}	9×10^{-20}
3	1×10^{-11}	4×10^{-19}	8×10^{-10}	2×10^{-19}
4	5×10^{-11}	9×10^{-19}	1×10^{-9}	4×10^{-19}
5	6×10^{-11}	3×10^{-18}	3×10^{-9}	1×10^{-18}
6	3×10^{-10}	1×10^{-17}	9×10^{-9}	5×10^{-18}
7	2×10^{-9}	5×10^{-17}	7×10^{-8}	2×10^{-17}
8	3×10^{-8}	2×10^{-16}	1×10^{-7}	1×10^{-16}
9	2×10^{-7}	1×10^{-15}	6×10^{-7}	5×10^{-16}
10	2×10^{-6}	6×10^{-15}	4×10^{-6}	2×10^{-15}
11	5×10^{-6}	3×10^{-14}	2×10^{-5}	1×10^{-14}
12	1×10^{-5}	1×10^{-13}	7×10^{-5}	6×10^{-14}
13	4×10^{-5}	7×10^{-13}	5×10^{-4}	3×10^{-13}
14	3×10^{-4}	4×10^{-12}	3×10^{-3}	1×10^{-12}
15	1×10^{-3}	2×10^{-11}	7×10^{-3}	8×10^{-12}
16	1×10^{-2}	1×10^{-10}	2×10^{-2}	5×10^{-11}
17	5×10^{-2}	5×10^{-10}	3×10^{-2}	6×10^{-10}
18	4×10^{-1}	3×10^{-9}	3×10^{-1}	4×10^{-9}

We found that the 12-th degree approximation provides sufficient accuracy for our algorithms and such invariants can be successfully deployed for our purposes. We however came across some symbols that after normalization exhibit a substantial maximum absolute error. These are linear characters, such as "-", "/", "l", etc. Integral invariant of the first order of these symbols is close to identical zero (Figure 3.2). Therefore, normalization and approximation are not stable.

These kinds of samples tends to behave unpredictably under affine transforma-

tions. They can, essentially, be transformed into anything if stretched in the direction orthogonal to the line. Small symbols are affected by scale normalization in a similar way (a resized period or comma can also look like a different character).

Chapter 4

Rotation-Invariant Recognition

Rotation transformations are very common in handwriting, as it is shown in Figure 4.1. This chapter is based on the paper "Orientation-independent recognition of handwritten characters with integral invariants" [28] co-authored with Golubitsky and Watt. We propose a rotation-invariant classification algorithm and test its performance on our dataset of handwritten samples.

4.1 Coefficients of Coordinate Functions and Integral Invariants

Curve parameterization and moments are computed online, while the curve is being written. Approximation of the stroke is recovered from its moments [35] in a small overhead that is quadratic in the degree of approximation. We observed that the 12-th degree Legendre-Sobolev series gives an approximation that is almost indistinguishable from the original curve.

Having computed coefficients of approximation of coordinate functions $X(\lambda)$ and $Y(\lambda)$:

$$(x_0, x_1, ..., x_d, y_0, y_1, ..., y_d),$$

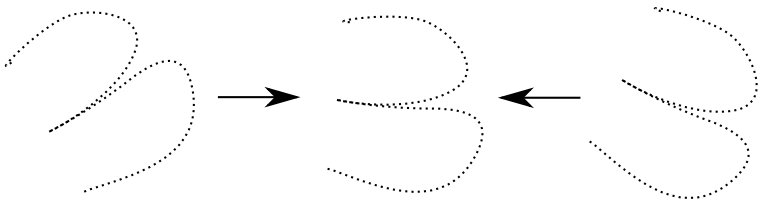


Figure 4.1: Rotation of a symbol

invariants are written as in Section 3.2. Taking advantage of orthogonality of Legendre-Sobolev polynomials, approximation of these invariants is calculated as

$$I_{k,i} = \frac{\langle I_k(\lambda), P_i(\lambda)\rangle}{\langle P_i(\lambda), P_i(\lambda)\rangle} \quad i = 1..d, k = 0..2.$$

These coefficients represent a sample in a compact and descriptive way and also serve as the basis for possible transformations of the character, such as morphing.

We represent the formula for I_1 as in Section 3.2 to be able to precompute the second term for corresponding polynomials and store it in a file to accommodate for the frequent use. This way, it takes cubic time to approximate I_1. Similarly, we compute each coefficient of I_2 in $O(d^4)$ operations. Since approximation with polynomials of degree 12 gives sufficient accuracy, coefficients of invariants are computed fairly quickly. If desired, approximation of invariants of higher degrees can be calculated. However, we expect those to improve recognition only slightly, while considerably increasing the computational overhead. For example, it would require $O(d^7)$ operations to compute coefficients of I_3.

4.2 Classification with Integral Invariants

We assume to have computed and normalized the coefficients of invariants of a given character

$$(\bar{I}_{0,1}, ..., \bar{I}_{0,d}, \bar{I}_{1,1}, ..., \bar{I}_{1,d}).$$

The sample is classified based on the distance to convex hulls of nearest neighbours

in the space of Legendre-Sobolev coefficients of integral invariants, as described in Section 2.4. Even though this method has high invariance to rotation, we expect it to perform poorly, because invariants are not as good curve descriptors as the coordinate functions.

4.3 Classification with Coordinate Functions and Integral Invariants

An improved version of CII is the classification with coordinate functions and integral invariants (CCFII). In this algorithm, coefficients of approximation of invariants are computed to select N classes with convex hulls located on the closest distance to the subject sample. The value of N is determined experimentally to ensure a high probability of including the correct class. For each of the N classes, we evaluate the minimal distance with respect to the sample rotation. In other words, for each of the classes, we solve the minimization problem to find the angle of rotation and correct class

$$\min_{\alpha}\left(\sum_{k}(X_k-(x_k\cos\alpha+y_k\sin\alpha))^2+\sum_{k}(Y_k-(-x_k\sin\alpha+y_k\cos\alpha))^2\right)$$

where x_k, y_k are the coefficients of approximation of coordinate functions of the test sample, and X_k, Y_k are that of a training symbol.

The solution to the problem is selected among the values of the function at boundary points of the closed interval of rotation and at the stationary point

$$\alpha=\arctan\left(\frac{\sum_k(X_ky_k-Y_kx_k)}{\sum_k(X_kx_k+Y_ky_k)}\right).$$

4.4 Classification with Coordinate Functions and Moment Invariants

Moment invariants were described in Section 2.7. Classification with Coordinate Functions and Moment Invariants (CCFMI) is similar to CCFII, except for the rotation invariant function. We represent the $(p+q)$-th order moment as

$$m_{pq}(\lambda_\ell) = \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} X(\lambda_i)^p Y(\lambda_j)^q f\big(X(\lambda_i), Y(\lambda_j)\big)$$

where $X(\lambda_i)$ and $Y(\lambda_i)$ are coordinates of the i-th point of the sample.

We take f of the form $f\big(X(\lambda_i), Y(\lambda_j)\big) = \sqrt{X(\lambda_i)^2 + Y(\lambda_j)^2}$. Considering that the size and position of a sample are already normalized in our algorithms, we work with moments directly to build the following rotation invariants

$$M_0(\lambda) = m_{00}(\lambda),$$
$$M_1(\lambda) = m_{20}(\lambda) + m_{02}(\lambda),$$
$$M_2(\lambda) = (m_{20}(\lambda) - m_{02}(\lambda))^2 + 4m_{11}(\lambda)^2.$$

Similar to the method described in the previous section, this algorithm selects top N candidates with the moment invariants.

4.5 Evaluation of Results

We evaluated performance of the algorithms for two combinations of integral invariants: $I_0(\lambda)$, $I_1(\lambda)$ and $I_0(\lambda)$, $I_1(\lambda)$, $I_2(\lambda)$, and similarly for moment invariants. Presence of the second-order invariant in the classification algorithm gives only a minor improvement in recognition rate – about 1% – while introducing the complexity of computing coefficients of approximation of $O(d^5)$, where d is the dimension of the

Table 4.1: Presence of the correct class within the top N classes, CCFII

$N=1$	2	3	4	5	6	7	10	15	**20**	25
87.9	95.1	96.8	97.7	98.3	98.7	98.9	99.4	99.5	**99.5**	99.5

Table 4.2: Error rate (%) for different numbers of nearest neighbours, CCFII

angle (radians)	$K=8$	10	12	14	16	18	19	**20**	21	22
0	4.4	3.9	4.2	4.0	3.9	3.9	3.8	**3.7**	3.8	3.8
0.3	6.2	5.7	5.7	5.4	5.4	5.4	5.3	**5.3**	5.4	5.4
0.5	7.4	6.9	6.8	6.7	6.6	6.5	6.4	**6.4**	6.5	6.5
0.7	8.5	7.9	7.7	7.6	7.4	7.4	7.2	**7.2**	7.3	7.4
0.9	9.3	8.8	8.6	8.3	8.2	8.2	8.2	**8.1**	8.2	8.2
1.1	9.6	9.0	8.7	8.6	8.4	8.4	8.2	**8.2**	8.4	8.4
average	7.5	7.0	7.0	6.8	6.6	6.6	6.5	**6.5**	6.6	6.6

vector space. Therefore, for the purpose of rotation invariant recognition, we focus on $I_0(\lambda)$ and $I_1(\lambda)$.

Difference in classification results for moments M_0 and M_1 vs. M_0,M_1 and M_2 is similar to integral invariants. Therefore, for the fair comparison of performance, we chose to classify samples with I_0, I_1 vs. M_0, M_1 for different combinations of number of classes (N) and number of nearest neighbours (K).

Recognition rate of CII is 88%. The performance of the algorithm does not depend on the angle to which test samples are rotated, since coefficients of approximation of invariants are almost identical. The frequency of occurrence of the correct class in the top N classes is also independent of rotation angle. The classification rate of 88% is relatively low and can be explained, perhaps, by the fact that the first integral invariant do not conclusively describe a curve.

We observed recognition rate of CCFII to be significantly higher, since this algorithm includes coordinate functions analysis. To estimate the best classification rate of this algorithm, we first empirically found the number N of top classes that has

Table 4.3: Presence (%) of the correct class within the top N classes, CCFMI

$N=1$	2	3	4	5	10	20	30	40	**50**	55
51.5	68.3	77.2	82.2	85.9	95.3	98.8	98.9	99.0	**99.0**	99.0

high probability of containment of the correct class. Presence of the correct class in N for different N is given in Table 4.5. We take $N = 20$ as the appropriate balance between accuracy and time complexity introduced by integral invariants.

With fixed N, the relationship between the number of nearest neighbours K and error rate for different angles is shown in Table 4.5. The value of K which yields the highest recognition rate, was determined to be 20. With fixed K and N, the classification rate of CCFII is 96.3% for non-rotated samples. The rate has a minor decrease with the increase of rotation angle, but never approaches CII (see Table 4.5).

Classification rate of CCFMI was estimated in a similar way, although we had to take a relatively large number of classes, $N = 50$, to achieve an acceptable presence of the correct class. Classification error for this N is given in Table 4.5. Overall performance of moment invariants is worse than that of integral invariants, even when they require more computations (greater amount of classes and nearest neighbours in each class). The results of experiments to select the value of N for CCFII and CCFMI are plotted in Figure 4.3. Error rate for different K is shown in Figure 4.4. For the purpose of comparison, overall classification results are presented in Table 4.5 and in Figure 4.2. From the obtained results, we conclude that integral invariants are a suitable instrument for classification of handwritten characters when a deviation in orientation takes place and they outperform moment invariants.

Table 4.4: Error rate (%) for different numbers of nearest neighbours, CCFMI

angle (radians)	$K = 8$	10	12	14	16	18	20	21	**22**	23
0	7.0	6.6	6.4	6.2	6.1	6.1	5.9	5.8	**5.8**	6.0
0.3	8.0	7.8	7.6	7.4	7.2	7.1	7.0	7.2	**7.1**	7.2
0.5	9.3	9.1	8.9	8.5	8.3	8.3	8.2	8.2	**8.1**	8.3
0.7	10.4	10.1	9.9	9.5	9.4	9.2	9.1	9.2	**9.2**	9.3
0.9	11.5	11.1	10.8	10.4	10.2	10.2	10.1	10.0	**10.0**	10.0
1.1	11.4	11.1	10.7	10.4	10.2	10.1	10.1	10.0	**10.0**	10.0
average	9.6	9.3	9.1	8.7	8.6	8.5	8.4	8.4	**8.4**	8.5

Table 4.5: Error rates of CII, CCFII and CCFMI

α, rad.	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	1.0	1.1
CII	12	12	12	12	12	12	12	12	12	12
CCFII	**3.7**	**3.9**	**4.5**	**5.3**	**5.9**	**6.4**	**6.6**	**7.2**	**8.2**	**8.2**
CCFMI	5.8	5.9	6.5	7.1	7.7	8.1	8.7	9.2	10	10

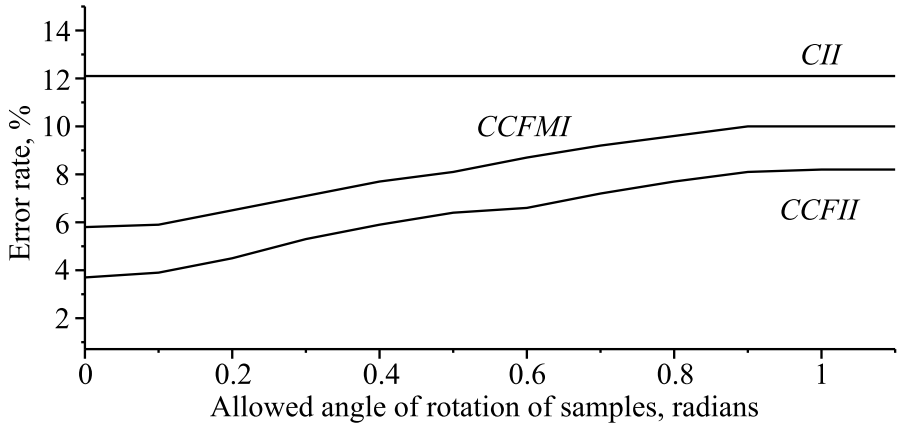


Figure 4.2: Error rates of CII, CCFII, CCFMI

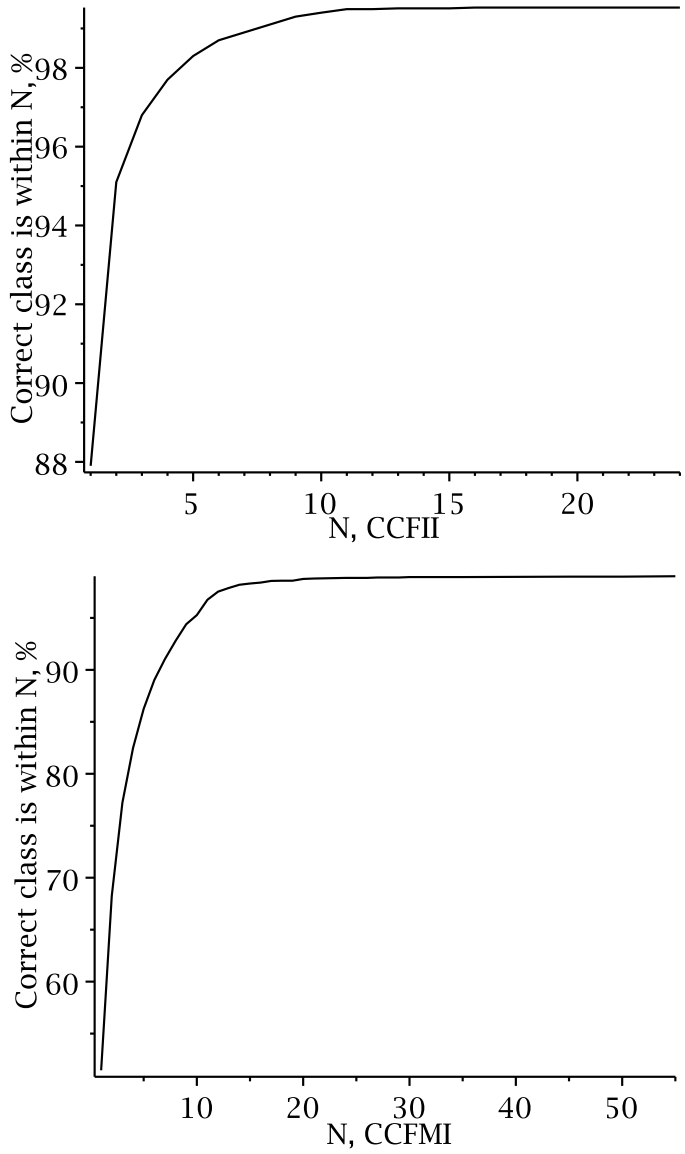


Figure 4.3: Presence of the correct class within N for CCFII (top) and CCFMI (bottom)

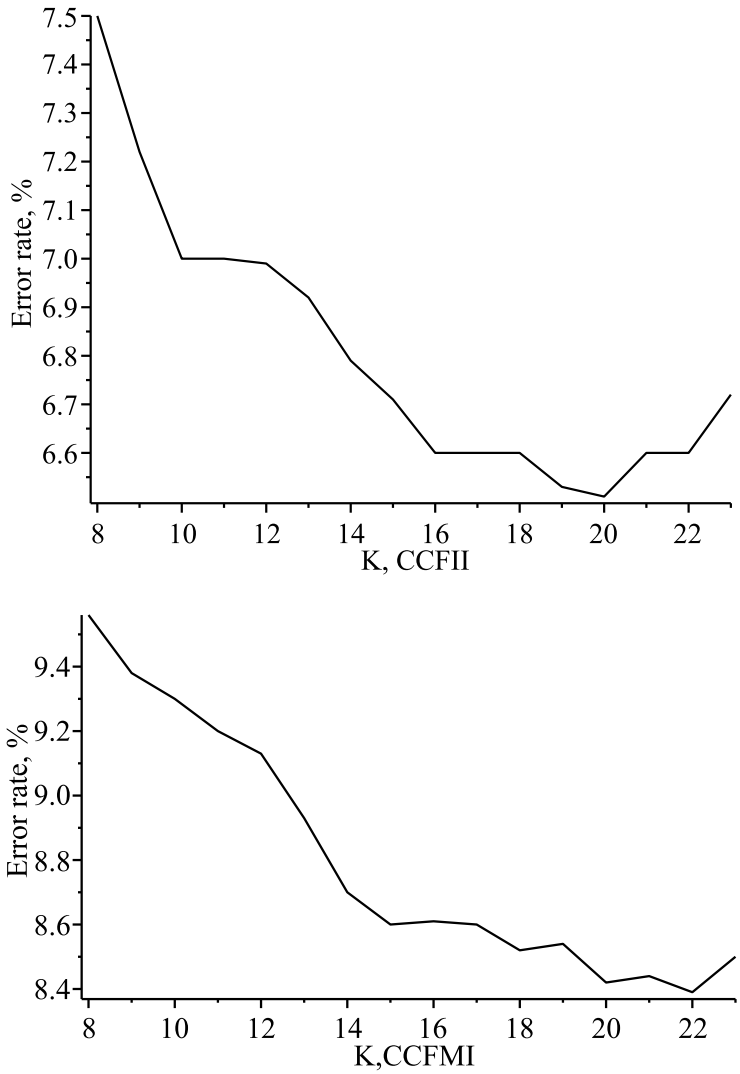


Figure 4.4: Error rate for different K for CCFII (top) and CCFMI (bottom)

4.6 Summary

We presented methods to classify handwritten characters, independently of orientation, based on integral invariants. We compared performance of integral invariants with geometric moment invariants. We observed that integral invariants perform better and require less computation. We therefore conclude that integral invariants are a suitable instrument in the recognition of handwritten characters when orientation is uncertain.

As expected, we noticed an increase in error rate with the rotation angle for CCFII and CCFMI. The typical misclassification is when distinct symbols have similar shape and are normally distinguished by their orientation, for example "1" and "/", "+" and "×", "U" and "⊂". As a possible solution, a system could consider a tendency to write characters in a similar orientation and restrict the range of angles for nearby symbols. A technique similar to CCFII can be applied to classify symbols as part of an expression with small adjustments to the minimization function.

Chapter 5

Shear-Invariant Recognition

This chapter addresses other form of transformation that commonly occurs in handwriting: shear or skew transformations. The chapter is based on the paper "Toward Affine Recognition of Handwritten Mathematical Characters" [29] co-authored with Golubitsky and Watt. In simple terms, shear invariance can be described as the process of "de-slanting" handwritten samples. In addition, we observed that the maximal shear angle, to which a character can be transformed and still remain recognizable by a human, can be relatively large (Figure 5.1), compared to the corresponding maximal rotation angle. We therefore consider shear as the transformation that may happen very frequently in practice and find shear invariance as a valuable addition to our classification algorithm. At the same time, shear is harder to treat than rotation. Due to the fact that shear does not preserve the length of strokes, the arc length parameterization is no longer suitable. Scale normalization should be handled differently as well. We address these issues and other related aspects in this chapter.

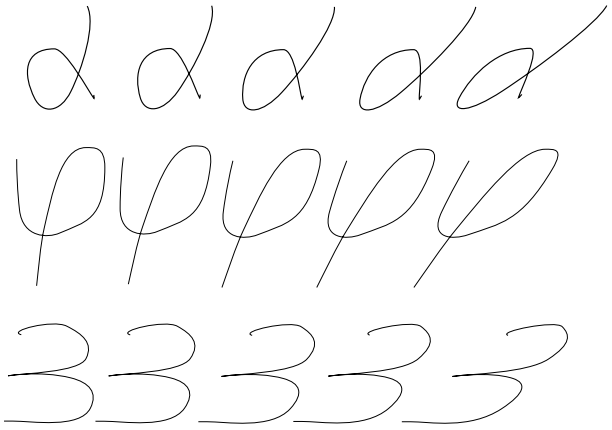


Figure 5.1: Different levels of skew of samples, from 0.0 to 0.8 radians with step of 0.2

5.1 Size Normalization

In most of the algorithms analyzed, size normalization is implemented by rescaling a character to achieve standard values of certain parameters, such as the Euclidean norm of the vector of Legendre-Sobolev coefficients of the coordinate functions [14]. While this approach can still be used in the case of rotated symbols [28], it is not applicable if samples are subjected to shear and affine transformations in general, since the arc length is not invariant under these distortions. To overcome this problem, we compute the norm $\|I_1\|$ of the Legendre-Sobolev coefficient vector of I_1. Coefficients of the coordinate functions are normalized by multiplying them by $1/\sqrt{\|I_1\|}$. Then, I_2 and its approximation can be computed from the normalized coefficients of the coordinate functions. Computing the norm of I_1 allows to extend the invariance of I_1 and I_2 from the special linear group, $SL(2, R)$, to the general linear group, $GL(2, R)$. Invariance under the general affine group, $\mathrm{Aff}(2, R)$, is obtained by dropping the first (order-0) coefficients from the coefficient vectors of the coordinate functions [14].

Other approaches exist for scale normalization of objects in pattern recognition, e.g. normalization by height and by aspect ratio [3]. Most of such techniques have certain drawbacks, however. To consider the normalization by height, it is invariant under horizontal shear, but dependent on rotation. The aspect ratio parameterization

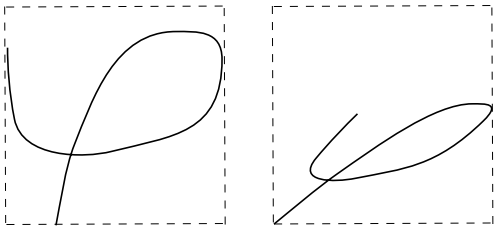


Figure 5.2: Aspect ratio size normalization

is robust under rotation, but inefficient for a noticeable degree of shear (Figure 5.2). To objectively evaluate the performance of our approach, we compare these techniques with the proposed normalization of taking the norm of $\|I_1\|$.

5.2 Parameterization of Coordinate Functions

Algorithms for online pattern recognition commonly deploy time or arc length as a parameterization of the coordinate functions because of its robustness and simplicity. The parameterization by arc length is of special interest in handwriting recognition, since it is not dependent on the local variations in speed of writing and is invariant under the group of Euclidean transformations. It may be expressed as

$$AL(\lambda) = \int_0^\lambda \sqrt{(X'(\tau))^2 + (Y'(\tau))^2}\mathrm{d}\tau.$$

Even though Euclidean transformations represent an important group of transformations that need to be considered, arc length does not perform that well for general affine transformations. Arc length, by definition, is not invariant under those transforms. Instead, we consider special affine arc length, which is invariant under the transformations we study and has been shown to perform well in pattern recognition [27]. We take the special affine arc length as

$$AAL(\lambda) = \int_0^\lambda \sqrt[3]{|X'(\tau)Y''(\tau) - X''(\tau)Y'(\tau)|}\mathrm{d}\tau.$$

In our experiments, we study recognition rate using each of the three parameterizations to evaluate performance empirically for different levels of distortion.

5.3 Shear-Invariant Algorithm

We assume to have computed coefficients of approximation of the coordinate and invariant functions. The proposed algorithm first selects N classes that are close to the test sample in the space of Legendre-Sobolev coefficients of integral invariants of the second and third order. Coefficients of the invariant and coordinate functions are computed as it was discussed in Section 4.1. Selection of the nearest neighbours and classes, based on the distance to convex hulls, is performed as described in Section 2.4.

Similar to the algorithm described in Section 4.3, we take N equal to 20. To select the correct class among the closest 20 classes, for each of these classes C_i, we solve the following minimization problem:

$$\min_{\phi} \mathrm{CHNN}_k(X(\phi), C_i),$$

where $X(\phi)$ is the sheared image of the test sample X and $\mathrm{CHNN}_k(X, C)$ is the distance from a point X to the convex hull of k nearest neighbours in class C.

Considering the small amount of candidate classes, the minimization problem can be solved by computing the distance for all possible angles with the step of 1 degree. This is the approach we used in our experiments. Nevertheless, there are more efficient ways. One could represent a class C as a single point $(X_0, \ldots, X_d, Y_0, \ldots, Y_d)$ in the Legendre-Sobolev space of the coordinate functions, and then find the minimum distance values at the boundary points of the interval of shear and the stationary point

$$\varphi = \arctan \frac{\sum_k (X_k - x_k)}{\sum_k y_k}.$$

Furthermore, as the curve is sheared by different angles, the corresponding point in the Legendre-Sobolev space traces a straight line segment. This follows from the fact that, for each order i, the Legendre-Sobolev coefficient x_i remains unchanged, while y_i is multiplied by $t = \tan(\phi)$, which spans the interval $[\tan(\phi_{\min}), \tan(\phi_{\max})]$. Therefore, we are looking at the problem of finding the point in the segment with the smallest CHNN distance. Furthermore, the task can be considered as finding the distance between two convex polyhedra. Taking into account that one of the polyhedra is just a segment, a number of consecutive optimizations can be considered.

5.4 Evaluation of Results

Classification rate of the algorithms has been evaluated for the discussed choices of parameterization of the coordinate functions: arc length (AL), time and affine arc length (AAL). Performance of the size normalization techniques has been tested as well (Figure 5.3 and Figure 5.4).

The parameterization by affine arc length makes the classification algorithm perform relatively poorly. A possible explanation is the presence of the second order derivative in the definition, which makes it sensitive to sampling perturbations. Therefore, the curve description becomes fuzzy, even though this parameterization is invariant under special affine transformations. The parameterization by arc length traditionally performs well on non-distorted samples. When an affine transformation (other than orthogonal) takes place, the classification ate drops considerably. Nevertheless, for shear up to 0.45 radians ($\approx$25 degrees) the parameterization by arc length gives noticeably better classification rate than time due to the intrinsic property of arc length being independent of speed of writing.

Higher invariance of the parameterization by time, compared to the parameterization by arc length, for large affine transformations is quite predictable. It can

be explained by time being invariant under any transformations of sampling points, while arc length is independent only under actions of rotation, scaling and translation (among the transformations that we are interested in). On the example of vertical shear, horizontal parts of a curve get stretched, while vertical parts remain unaltered. Therefore, arc length is subjected to considerable distortion.

Previous results showed [14] that arc length performs better when affine noise is of minor degree. We obtained similar results in our experiments, as it is presented in Tables 5.1(a)–5.1(d). However, we also see that time becomes more robust for bigger transformations. This can be explained by the fact that the timing of points on the curve is noisy in general, since various users typically write the same sample with different speed. In summary, the arc length signal is invariant with respect to variations in speed, but gets blurred by affine distortions; and it is opposite for the time parameterization. This leads us to the question of whether time and arc length can be combined in a parameterization that would inherit the advantages and, simultaneously, avoid the drawbacks of both.

Since time is mainly affected by local variations in speed of writing, one should expect that the effect of these variations, accumulated over longer time periods, will be close to neutral. The situation is opposite for arc length: large distortions become noticeable on bigger intervals. Essentially, parameterization by arc length is more accurate locally, while parameterization by time is more reliable globally. This allows us to combine these parameterizations as follows: divide the curve in N equal time intervals, and parameterize each interval by arc length. However, if this algorithm is implemented directly, an unexpected behaviour may occur at the end points of the time intervals. This can be prevented by smoothing the transition from time to arc length inside the subintervals with this form of mixed metric

$$kdt^2 + dx^2 + dy^2.$$

In the above formula, pure arc length would correspond to $k = 0$. To make the classification rate of the algorithm even higher, we take the number of classes selected by invariants to be 50 and find the optimal values of the mixed metric parameters $N = 2$ and $k = 2$ by cross-validation (see Table 5.2). We perform a similar experiment on the subset of LaViola dataset [19], with the results shown in Table 5.3. The obtained results are plotted in Figures 5.5. Therefore, we were able to make the parameterization just as invariant under transformations as time, which allows to describe the curve as well as arc length.

Our results show that the size normalization by height yields the best classification rate under shear transformation (Table 5.1(a)). This can be explained by the fact that horizontal shear preserves the height of samples. Height however is not an appropriate norm for some other affine transformations, e.g. rotation. Aspect ratio normalization (Table 5.1(b)) gives an acceptable performance rate for small degrees of shear distortion. However, the rate starts to decrease significantly for degrees of shear greater than 0.7 radians for parameterization by time and affine arc length. The proposed normalization by I_1 performs approximately as well as normalization by height (Table 5.1(c)), but has an advantage of being invariant under the full group of affine deformations. Therefore, we find the normalization of a character by the norm of I_1 to be the most robust method in the presence of transformations. However, in some cases I_1 can not be used as the norm. If a character has linear shape, the invariant of the first-order will be close to being identically zero, since it is the area between a curve and its secant, as in Figure 3.2. Taking the norm of I_1 in this case can make the symbol look like any other symbol, causing partial misclassification. We call such samples linear symbols and notice that they perform unpredictably with most of the techniques designed for two-dimensional curves. Essentially, under affine distortions, these samples can be transformed into anything, if stretched in the direction orthogonal to the line. A similar effect can be seen on small symbols subjected

Table 5.1: Recognition rate (%) for shear from 0.0 to 0.9 radians and for parameterization by affine arc length (AAL), arc length (AL) and time

(a) Size normalization by height

	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
AAL	82.2	82.2	82.2	82.1	82.1	82.1	82.1	82.1	82.1	82
AL	96.4	96.4	96.1	95.6	95	94.1	93	91.9	90.2	88
Time	94.8	94.9	94.9	94.7	94.5	94.4	94.4	94.4	94.4	94.3

(b) Aspect ratio size normalization

	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
AAL	81.9	81.8	81.6	81.4	81.2	81	80.8	80.2	79.4	77.4
AL	96.3	96.4	96.1	95.5	94.7	93.7	92.3	90.1	85.7	77.5
Time	94.7	94.7	94.6	94.3	94.1	93.9	93.7	93.2	91.9	89.

(c) Size normalization by I_1

	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
AAL	83	83.1	83	82.9	82.9	82.8	82.8	82.8	82.8	82.7
AL	**96.3**	**96.3**	**96.1**	**95.7**	**95.1**	**94.4**	**93.3**	**91.9**	**90.2**	**87.9**
Time	94.6	94.7	94.6	94.5	94.5	94.5	94.5	94.5	94.5	94.4

(d) Size normalization by I_1, without linear symbols

	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
AL	**96.7**	**96.7**	**96.5**	**96.2**	**95.7**	**95.0**	**94.1**	**92.7**	**91.2**	**88.9**

to size normalization – a bloated dot or comma can also appear as a completely different character. To prove this hypothesis, we performed experiments without linear symbols. We excluded 1,442 samples, or 3% of the total number of samples, such as "-", "\", "/", "l", ".". As the result, performance improved by about 0.6% for corresponding degrees of skew (Figure 5.4). The obtained improvement leads us to the conclusion that a robust classification method should be able to detect such linear symbols and treat them in a special way.

5.5 Toward Unified Affine-Invariant Classification

We have shown in Section 4 how to recognize symbols independently of rotation. To achieve the best classification rate, one could apply both of these techniques to a

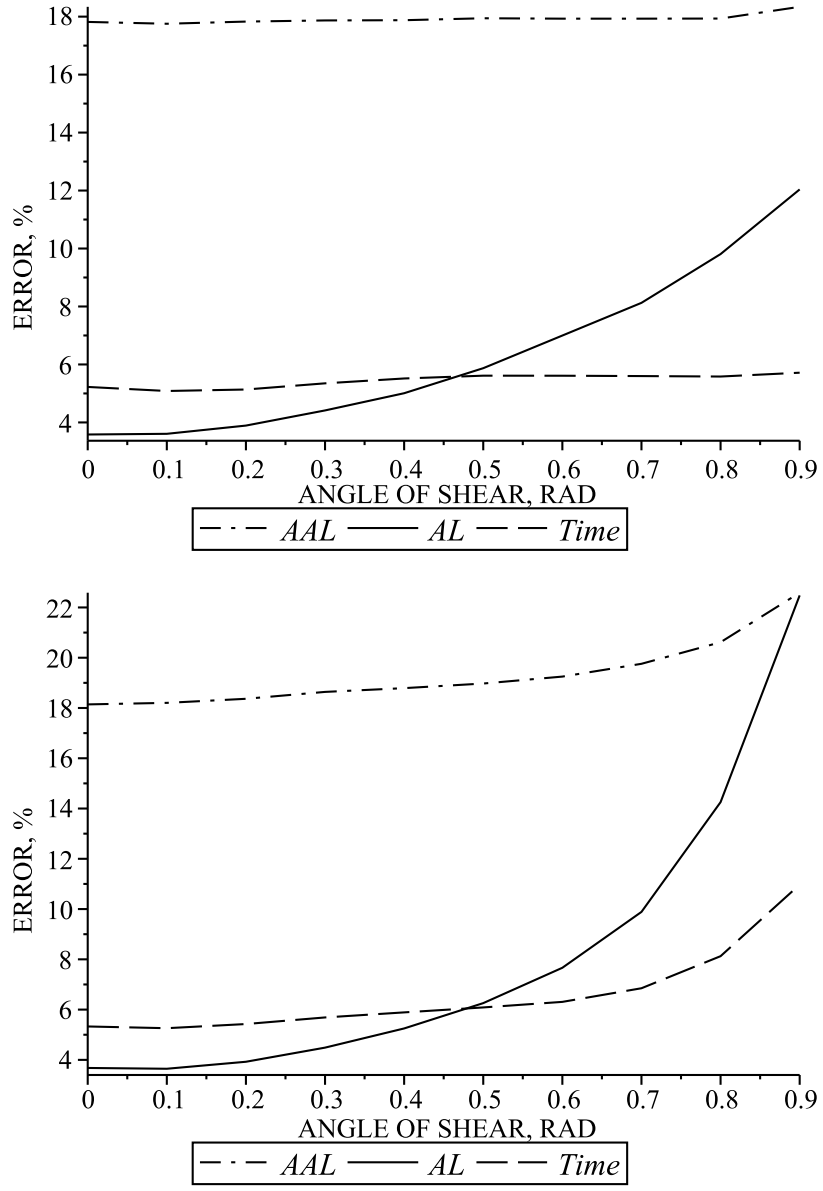


Figure 5.3: Error rate for size normalization by height (left) and with aspect ratio (right)

Table 5.2: Recognition rate (%) for mixed parameterization for corresponding values of N and k

	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$N = 2$, $k = 1$	96.1	96.2	96	95.9	95.8	95.7	95.6	95.3	95.2	94.9
$N = 2$, $k = 2$	95.8	95.9	96	95.8	95.7	95.7	95.7	95.6	95.5	95.5
$N = 3$, $k = 0.5$	96.1	96.2	96	95.9	95.9	95.7	95.5	95.6	95.3	95
$N = 3$, $k = 1$	95.9	96.2	96	95.8	96	95.8	95.7	95.6	95.5	95.2
$N = 4$, $k = 1$	95.9	96.1	95.9	95.7	95.8	95.6	95.6	95.7	95.6	95.4

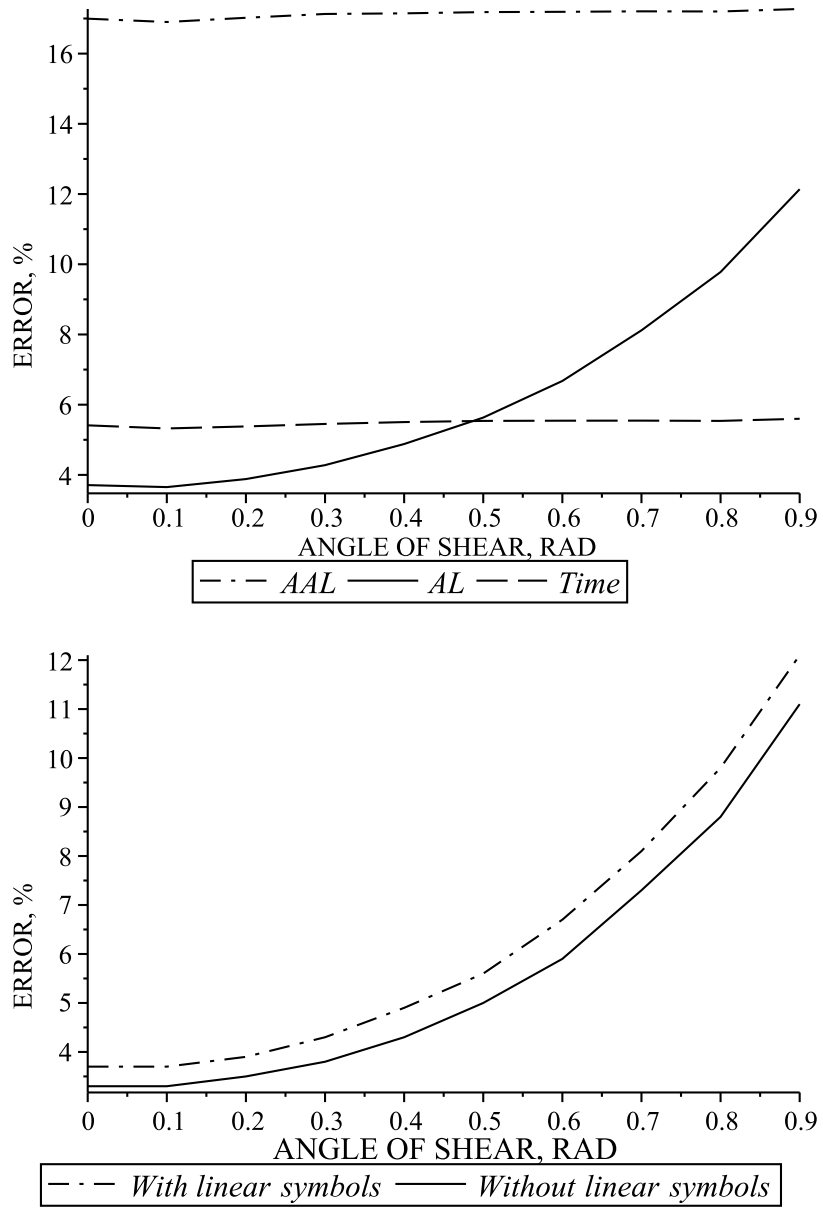


Figure 5.4: Error rate for size normalization with I_1 (left) and comparison of performance without linear samples (parameterization by arc length and size normalization with I_1 (right)

Table 5.3: Recognition rate (%) for mixed parameterization for LaViola component.

	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
$N = 2$, $k = 1$	98.4	98.5	98.5	98.4	98.3	98.4	98.3	98.3	98.2	98.0
$N = 2$, $k = 2$	98.4	98.5	98.4	98.3	98.4	98.3	98.3	98.3	98.2	98.2
$N = 3$, $k = 0.5$	98.3	98.3	98.3	98.2	98.3	98.2	98.3	98.2	98.1	98.0
$N = 3$, $k = 1$	98.4	98.4	98.4	98.4	98.4	98.4	98.3	98.3	98.3	98.2
$N = 4$, $k = 1$	98.4	98.5	98.4	98.4	98.3	98.3	98.3	98.3	98.2	98.2

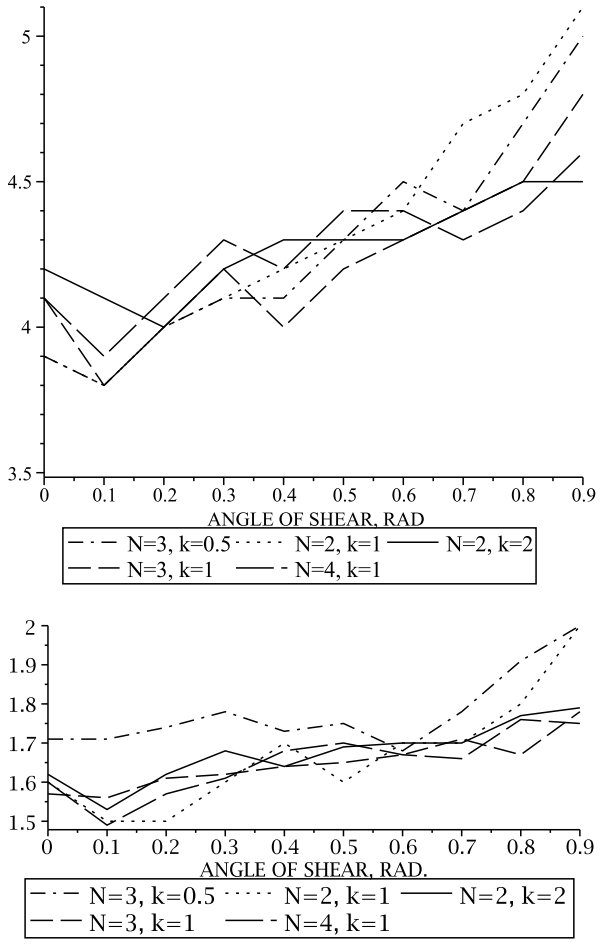


Figure 5.5: Error (%) for the mixed parameterization for different values of skew

sample. Such a minimization problem includes two parameters: degree of horizontal shear and degree of rotation. The vertical shear can be omitted from the analysis, since it is not common in handwriting. This algorithm will be invariant with respect to shear, rotation, scale and translation and should yield classification rates similar to those presented in Table 5.1(c) for parameterization by time and arc length or as in Table 5.2 for the mixed parameterization.

There is, however, an alternative to make an algorithm stable under transformations from the general affine group. Since I_1 performs poorly as a norm when it is close to zero, i.e., on linear and small symbols, the appropriate representation should combine normalized and non-normalized features. Perhaps one could put the size of a

sample, the Legendre-Sobolev coefficients of the coordinate functions, and Legendre-Sobolev coefficients of the invariants together in a feature vector. Each of these three parts should be scaled by weight.

The weights can be symbol-dependent, such that for a small symbol, size should be weighted more, while the rest of the vector would weight less. For a linear symbol, size and Legendre-Sobolev coefficients of the affine invariants should weight less, while coefficients of the coordinate functions should be weighted more. For regular non-linear and non-small symbols, size should be weighted less, and the Legendre-Sobolev coefficients of the coordinate functions and invariants can be weighted similarly.

The question of how to calculate the weights prior to classification is important. If one doubles the size of a period or a comma, it may not appear as such anymore. But if size of a sample, such as "O", is doubled, it would still clearly be "O". Similarly, if a non-linear symbol is stretched in any direction by some factor, in most cases, it would be classified as the same symbol. Therefore, we can consider a measure of "affinity" of a symbol with respect to a transformation group, as the degree to which the symbol can be transformed while still appearing as the same symbol.

5.6 Summary

We have presented a classification algorithm that is invariant under shear, rotation, scaling and translation, the subset of the most important affine transformations in recognition of handwriting.

To achieve independence of scale, we evaluated popular size normalization methods, but concluded that they are not suitable for the affine case. Instead, we perform size normalization by normalizing the coefficient vector of the integral invariant of the first order.

Another challenge was parameterization of coordinate functions. Parameterization

by time gives stable recognition rate for different levels of distortion, while arc length performs noticeably better for small transformations. To take advantage of both of these parameterizations we constructed a new mixed parameterization, by dividing the curve into equal time intervals and parameterizing each interval by the combined metric. We experimentally evaluated the parameters, finding those that make the mixed parameterization close to the results of parameterization by arc length, while being almost as invariant under distortions as parameterization by time. We have explained how to change the minimization problem to include rotation invariance. Finally, we have laid out a more general approach of extending the robustness to the full affine group for unified affine-invariant classification.

Part II

Compression of Digital Ink

Chapter 6

Compressed Storage of Handwriting

Naïve representation of digital handwriting suffers from excessive demand of storage, since every point of a curve is represented by a vector. Such a vector is at least of two dimensions, if only X and Y coordinates of a sample are stored. In a general case, every point may store other associated values, such as pen pressure, tilt, time, etc.

In the work on symbol recognition, presented in Chapters 4 and 5, we found it useful to represent digital strokes in a functional form, as coefficients of truncated a orthogonal series. This form, described in more detail in this chapter, represents curves quite succinctly. It is natural to ask whether this form may be used in compression. A great advantage of such representation, besides significant saving on storage space, is the ability to use compressed samples directly in the recognition algorithms. This chapter is based on the paper "Digital Ink Compression via Functional Approximation" accepted to the 12th International Conference on Frontiers in Handwriting Recognition, (ICFHR 2010), co-authored with Watt [25].

We take the view that lossless compression at time of ink capture is not a meaningful concept as each ink capture device has a resolution limit and sampling accu-

racy. As 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 result in symbols that a human reader would recognize as indistinguishable.

This chapter 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 lossy compression with functional approximation at device resolution gives about twice the compression rate obtained by the lossless method of compressing second differences of sample point coordinates.

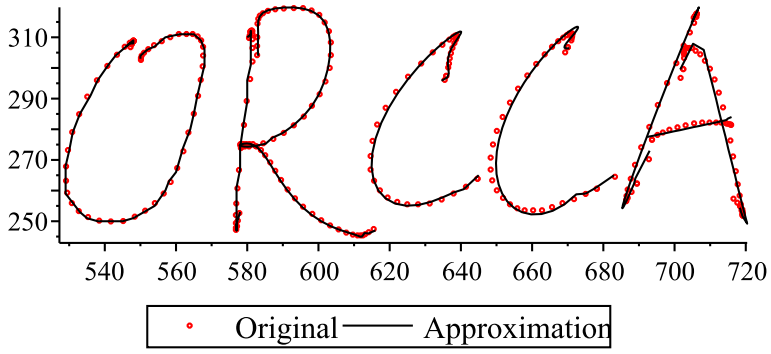
The rest of the chapter is organized as follows. Section 6.1 explains the problem we solve and gives an example of approximation of curves with different error limits. Section 6.2 describes the compression method. The experiments to evaluate the compression method are described in Section 6.3. Compression results for representing coefficients in text and binary formats are given in Sections 6.3.2 and 6.3.3 respectively. Evaluation of the second differences method and comparison with our approach is presented in Section 6.3.4. Section 6.4 concludes the chapter.

6.1 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 absolute error and root mean square error. If so, what is the compression one could expect as the result of such approximation?

We empirically investigate different approaches to obtain the minimal overall size of coefficients of an approximation that satisfies the given error constraints. We consider compression of handwritten regular text, since it commonly occurs in pen-based

Table 6.1: Approximation thresholds



	O	R	C	C	A
Max. abs. error, %	1	2	3	4	5
RMSE, %	0.33	0.67	1	1.33	1.67

computing and incorporates different kinds of patterns. An example of a word and its approximation with different thresholds are shown in Table 6.1 and the corresponding figure. We take RMSE as a third of the maximum error and consider both thresholds to define the quality of approximation. To measure the quality of approximation independently of application and device, we compute errors as the percent of the height of characters in a stroke.

6.2 Algorithms

6.2.1 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 ($e.g.\ x, y, p$)
3. Deflate the stream of coefficients.

To reconstruct a stroke, the process is reversed:

1. Inflate 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.

For a given segment, the series coefficients are computed by numerical integration of the required inner products.

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 [40] and Huffman coding [17]. This is for convenience only – a more specialized method would be used in a production setting.

6.2.2 Parameterization Choice

We tested the two choices for parameterization of coordinate functions widely used in pen-based computing: time and arc length. We observed that parameterization by time, while being more efficient, gives better compression. Comparison of the results is presented in the Fig. 6.2 for approximation with Chebyshev polynomials with 0 size of the fractional part in coefficients.

6.2.3 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. As described below, we explored the three methods to segment traces.

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 Size Segmentation We fix the size 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 segment the curve with all combinations. 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.

6.2.4 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 Table 6.1. 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

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

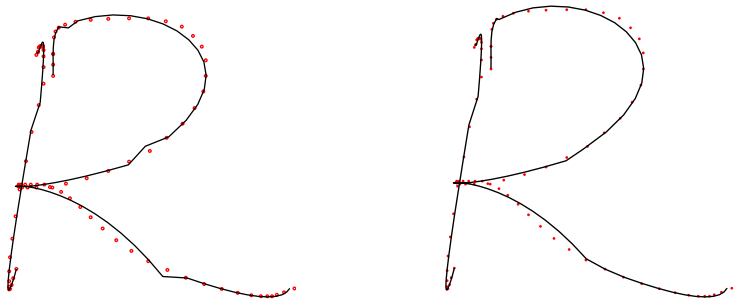


Figure 6.1: Example of blending

with the weight function

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

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 may have its own portion of overlap necessary for smooth transition. An example of a blended sample is given in the Fig. 6.1.

6.3 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.

6.3.1 Experimental Setting

The dataset of handwritten samples was collected in the Ontario Research Center for Computer Algebra with various tablet devices. Specifications of the device are: 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. The error, relative to the height of writing, is $\approx 2.5\%$.

Several 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%.

6.3.2 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, but this is relevant for standard XML representations.

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

$$\lambda_0; c^1_{00}, c^1_{01}, ..., c^1_{0d_{01}}; ...; c^N_{00}, c^N_{01}, ..., c^N_{0d_{0N}}$$

$$\lambda_1; c^1_{10}, c^1_{11}, ..., c^1_{1d_{11}}; ...; c^N_{10}, c^N_{11}, ..., c^N_{1d_{1N}}$$

$$...$$

$$\lambda_D$$

where λ_i is the initial parameterization value of piece i in the stream, N is the number

of channels 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 $f_i^j(\lambda)$, $i = 0..(D-1)$, $j = 0..N$ 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 find 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 is how dependent the compression on the size of the fractional part of coefficients. The result of the experiment for Chebyshev polynomials for the *fixed degree* method for different fractional sizes is presented in Table 6.2 for different degrees of approximation for the maximal error of 3%. Taking the combination of the size of fractional part of 1 and degree 7, we find compressed size for other error thresholds. The same procedure was performed for other approximation methods (Fig. 6.3).

A similar experiment for Chebyshev polynomials has been performed for the *fixed length* method. Results are shown in Table 6.3, in which "–" denotes the case, when an interval exists, that can not be approximated, even with the orthogonal series of degree 20.

We observed that the fixed degree method performs significantly better and eliminates the risk associated with fixed length – the existence of an interval that can not be approximated. Therefore, all the consecutive experiments were performed with variations of the fixed degree method.

Table 6.2: 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\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 6.3: Compressed size (%) by length of intervals (L) and fractional size of coefficients (F) for Chebyshev polynomials and max. error of 3%, fixed length method

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	–

Fixing the Size of Coefficients In the next experiment we ask whether compression will change if we take coefficients as real numbers with fixed amount of digits. Similar to the previous experiment, we look at the rates for different degrees and the number of digits to find the optimal combination for maximal error of 3%. Taking this combination, we then find compression for other values of maximal and RMS errors. We however observed that applying this algorithm literally is not a preferred solution, since the coefficient of the 0-th degree polynomial usually significantly exceed the rest coefficients. This can be explained by the fact that the 0-th degree polynomial is 1 and its coefficient serves as a position translator. We therefore allow this coefficient to be twice of the size of coefficients of higher degree. In Table 6.4 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 methods is given on the Fig. 6.4.

Table 6.4: Compressed size (%) for different approximation degrees (D) and coefficient sizes (S) for Chebyshev polynomials with max. error of 3%, fixed degree 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

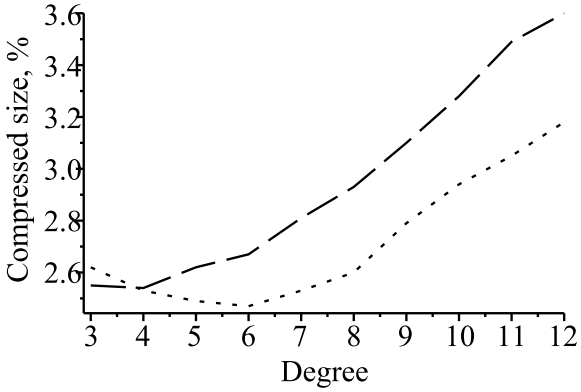


Figure 6.2: Compression for parameterization by time (dot) vs. arc length (dash) for different degrees of approximation, fractional part of size 0

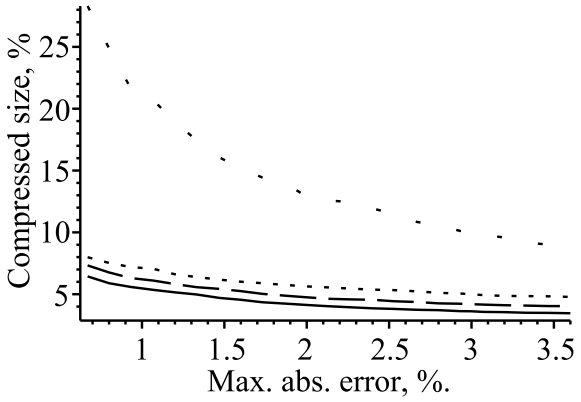


Figure 6.3: Compressed size for different values of error for Chebyshev (solid), Legendre (dash), Legendre-Sobolev (dot) and Fourier (space dot): coefficients with 1 digit fractional part

Table 6.5: Conversion matrix condition numbers for different degrees (D) for Legendre (L) and Legendre-Sobolev (L-S) bases

B\D	4	5	6	7	8	9	11	12
L	2.9	4	4.5	5.6	6	7	8.3	8.7
L-S	100	350.8	1394.1	5422.6	20064.3	71597.4	1042702.43	3942541.38

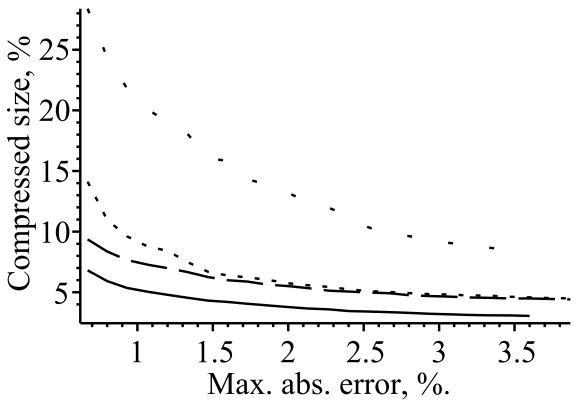


Figure 6.4: Compressed size for different values of error for Chebyshev (solid), Legendre (dash), Legendre-Sobolev (dot) and Fourier (space dot): coefficients with fixed coefficient size

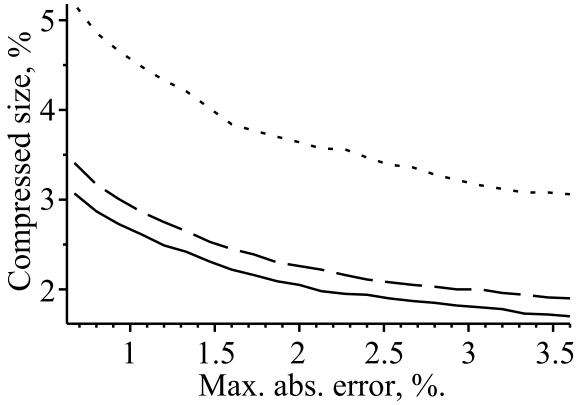


Figure 6.5: Compressed size for different values of error for Chebyshev (solid), Legendre (dash), Legendre-Sobolev (dot) and Fourier (space dot): coefficients with binary representation

6.3.3 Compression of Binary Traces

We now explore how to compress data for applications that can store ink data in binary form. To do this we store the sequence of approximation coefficients compactly in an exponential format as ab where a and b are two's complement binary integers, standing for significant and a power of 10 respectively. We fix the size of b to 3 bits and change only the size of a.

We note that the fixed degree and fixed length segmentation schemes have pa-

rameters which optimal choice depends on the application. Certain types of strokes have their own optimal combination of parameters. It becomes especially noticeable when curve patterns have completely different styles: from straight line to curly handwriting.

Therefore, for our final experiment we use the adaptive segmentation scheme and choose stroke-wise approximation parameters for each input channel separately. Essentially, compression packets for each stroke i take the form

$$\begin{aligned} & b_i; d_i; \lambda_1; c_{10}, c_{11}, ..., c_{1d_i} \\ & \lambda_2; c_{20}, c_{21}, ..., c_{2d_i} \\ & ... \\ & \lambda_D \end{aligned}$$

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 (Fig. 6.5, Tables 6.6 and 6.7): 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. Maximum error $< 2.5\%$ is indistinguishable by a human and such compression can be accepted as lossless for most of the applications in pen-based computing.

6.3.4 Comparison with Second Differences

The second differences method yields high compression for low-resolution devices and vice versa, assuming that sampling rate remains the same. A stroke is 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 store these values as binary numbers of fixed size, similar to the way it is described in Section 6.3.3. The size for the first two values is different

Table 6.6: Compressed size (%) for different errors (E) for representing coefficients in binary format for the lossless method of second differences ($\triangle 2$) and lossy compression with the following bases (B): Chebyshev (C), Legendre (L) and Legendre-Sobolev (L-S)

B\E,%	0.0	0.6	1.1	1.5	2.0	2.5	3.1	3.5
$\triangle 2$	23.35	–	–	–	–	–	–	–
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

Table 6.7: Compressed size (%) for different errors (E) for representing coefficients in binary deflated format for the lossless method of the second differences method ($\triangle 2$) and lossy compression with the following bases (B): Chebyshev (C), Legendre (L) and Legendre-Sobolev (L-S)

B\E,%	0.0	0.6	1.1	1.5	2.0	2.5	3.1	3.5
$\triangle 2$	8.64	–	–	–	–	–	–	–
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

from the size of second differences. In contrast with Section 6.3.3, we choose the number of bits to be able to store the maximum value, because of the nature of lossless compression. We then perform gzip encoding on this binary stream to model the compression algorithm described in [26]. For the handwriting collected with our device (Sec. 6.3), this method yields 8.64% compression.

The approach of representing handwriting by its approximation has an important advantage, other than better compression. It allows to build the database of handwritten samples and use it in recognition algorithms [14] 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.

6.4 Summary

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 [14] 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 [10]. One can gain the most advantage by storing compressed strokes represented by Chebyshev coefficients and converting them to Legendre or Legendre-Sobolev format by multiplication with the corresponding basis transformation matrix. The relationship between precision of coefficients of different bases is affected by the condition number of the conversion matrix. For conversion from Chebyshev to Legendre basis in the range of degrees 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 considered orthogonal bases. Another important aspect is to consider a different error measure – computing the Legendre-Sobolev distance allows us to estimate quality of approximation in the first jet space.

Chapter 7

Conclusion

7.1 Summary

The theory and experiments presented in this work contribute to the art of online recognition of handwritten characters and compression of digital ink. We represent written characters by coefficients of approximation of coordinate and invariant functions with orthogonal polynomials. This approach creates an opportunity to consider compression and recognition in a unified architecture for pen-based computing. The succinct representation of characters serves directly as a training input for classification algorithms.

We explored the recognition of handwritten characters subjected to rotation. To achieve invariance under this type of transformation, we deploy the theory of integral invariants. We represent a character by the coefficients of integral invariants of the coordinate functions. Since the invariants are chosen to be independent of area-preserving transformations, coefficients of the character remain the same when the sample undergoes arbitrary rotation. Classification is based on computing the distance to convex hulls of nearest neighbours in the space of coefficients of approximation of invariants. We observed, however, that coefficients of invariants alone have

relatively low recognition due to integral invariants being poor curve descriptors compared to coordinate functions. Therefore we perform additional analysis of subject samples by rotating them and evaluating the distance in the space of coefficients of coordinate functions. This algorithm has high recognition rate for both rotated and non-rotated samples.

To extend invariance of the method to general affine transformations, we had to solve the problem of size normalization and parameterization of coordinate functions. The robust size normalization approach introduced in our work involves taking the norm of coefficients of the integral invariant of the first order. This method allows extension of the first and second invariants from the special linear to the general linear group. Invariance with respect to translation is achieved by omitting the order-0 coefficient of approximation of the coordinate functions.

To implement an efficient parameterization of coordinate functions, we test the methods widely used in online pattern-recognition: time, arc length and affine arc length. We find these, however, to be either not invariant to affine transformations or to perform poorly. We propose a parameterization as a combination of time and arc length to inherit advantages and avoid drawbacks of both.

The proposed shear-invariant classification algorithm is similar to the rotation-invariant method – it selects a fixed number of candidate classes with coefficients of invariants and then analyses the selected classes with coefficients of coordinate functions. These methods result in high recognition rates and essentially insensitive to the most common transformations in handwriting.

To overcome the cumbersome representation of written samples by the coordinates of points, we propose to store coefficients of approximation instead. We develop a lossy compression scheme that allows us to compress handwriting with desired accuracy. We measure accuracy by the relative error, computed as the relation of the absolute error to the height of written samples. We note that lossy compression

is appropriate for our purposes, since digital handwriting is lossy by definition – points are collected at certain time intervals and traditional devices have sampling errors. We test different orthogonal bases for robust compression and find Chebyshev polynomials to be the most efficient choice. Examining the parameterization by time and arc length, we found the parameterization by time to perform slightly better and requiring less computations. We then propose a format of binary packets for compact representation of coefficients, in which the curve is split in segments. Each segment is represented by the combination of coefficients of corresponding coordinate functions. We study different ways of splitting a curve: Fixed Degree Segmentation, Fixed Size Segmentation and Adaptive Segmentation. The segment-wise approximation of a curve can leave strokes jaggy, especially for high error thresholds. We therefore propose a method of blending segments to avoid sharp edges at the boundaries of approximation. We highlight that the compression algorithm, presented in the work, has a higher deflation rate than the widely-adopted algorithm of compression with second differences of coordinates.

7.2 Future Work

For future work, we recognize the importance of developing an algorithm for affine invariant recognition of characters. Several such algorithms have already been proposed in this thesis, but are missing the necessary experimental verification. The next step is integration of the symbol classification techniques into a formula recognition framework. This task is not trivial, because of the 2-dimensional nature of mathematical characters. The key issue is careful differentiation between fluctuations in the positioning of written samples vs. intentional super- or sub-scripting.

The next interesting aspect is syntactic and semantic verification of recognized formulas. The main problem encountered is the absence of a fixed dictionary of

"words" or "set" of rules that controls the evolution of existing words. Therefore, algorithms based on learning from a given set of formulas are necessary for consecutive verification with logical programming or, possibly, with the theories of unification and anti-unification.

A possible future study regarding compression of a digital curve would consist of more detailed analysis of the tradeoff between the compression and recognition rates for a given orthogonal basis. We discovered that Chebyshev polynomials yield the best compression rate. Storing a stroke in Chebyshev coefficients and converting them to Legendre-Sobolev for the purpose of classification is computationally expensive and unreliable, since the condition number of a conversion matrix is exponentially dependent on the degree of approximation.

Integrating compression and recognition in one framework allows storage of hand-written samples compactly and can be efficiently used in recognition methods.

Bibliography

[1] S. Gilani A. Ali and N. Memon. Affine invariant contour descriptors using independent component analysis and dyadic wavelet transform. *Journal of Computing and Information Technology*, 16(3):169–181, 2008.

[2] R. K. Sharma Anuj Sharma, Rajesh Kumar. Online handwritten gurmukhi character recognition using elastic matching. In *Proceedings of the 2008 Congress on Image and Signal Processing*, volume 2, pages 391–396. IEEE Computer Society, May 2008.

[3] H. Sako C. Liu, K. Nakashima and H. Fujisawa. Handwritten digit recognition: investigation of normalization and feature extraction techniques. *Pattern Recognition*, 37:265–279, 2004.

[4] B.W. Char and S.M. Watt. Representing and characterizing handwritten mathematical symbols through succinct functional approximation. In *Proc. International Conference on Document Analysis and Recognition, (ICDAR)*, pages 1198–1202, Curitiba, Brazil, September 2007. IEEE Computer Society.

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

[6] Yi-Min Chee, Max Froumentin, and Stephen Watt. Ink markup language (InkML). `http://www.w3.org/TR/InkML/`. (valid on August 9, 2010).

[7] H. Sako C.L. Liu, K. Nakashima and H. Fujisawa. Handwritten digit recognition: Investigation of normalization and feature extraction techniques. *Pattern Recognition*, 37(2):265–279, 2004.

[8] Jon Ferraiolo, Fujisawa Jun, and Dean Jackson. *Scalable vector graphics (svg) 1.1 specification*. W3C, January 2003.

[9] Jan Flusser and Tomas Suk. Character recognition by affine moment invariants. In *Proc. of the 5th International Conference on Computer Analysis of Images and Patterns*, pages 572–577. Springer-Verlag, 2007.

[10] O. Golubitsky and S.M. Watt. Online stroke modeling for handwriting recognition. In *Proc. 18th Annual International Conference on Computer Science and Software Engineering, (CASCON 2008)*, pages 72–80, Toronto, Canada, October 2008. IBM Canada.

[11] O. Golubitsky and S.M. Watt. Online computation of similarity between handwritten characters. In *Proc. Document Recognition and Retrieval XVI, (DRR 2009)*, volume 7247, pages C1–C10, San Jose, California USA, January 2009. SPIE and IS&T.

[12] O. Golubitsky and S.M. Watt. Online recognition of multi-stroke symbols with orthogonal series. In *Proc. 10th International Conference on Document Analysis and Recognition, (ICDAR 2009)*, pages 1265–1269, Barcelona, Spain, July 2009. IEEE Computer Society.

[13] O. Golubitsky and S.M. Watt. Tie breaking for curve multiclassifiers. Technical report, Ontario Research Center for Computer Algebra, 2009. ORCCA TR-09-02.

[14] Oleg Golubitsky and Stephen M. Watt. Distance-based classification of hand-

written symbols. *International Journal of Document Analysis and Recognition*, (doi 10.1007/s10032-009-0107-7), 2010.

[15] Isabelle Guyon, Lambert Schomaker, Rkjean Planiondon, Mark Liberman, and Stan Janet. Unipen project of on-line data exchange and recognizer benchmarks. In *Proc. 12th International Conference on Pattern Recognition (ICPR 1994)*, pages 29–33, Jerusalem, Israel, 1994. IAPR-IEEE.

[16] M.K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, 1962.

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

[18] Qiang Huo and Tingting He. A minimax classification approach to hmm-based online handwritten chinese character recognition robust against affine distortions. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, pages 1226–1230. IEEE Computer Society, July 2007.

[19] Joseph J. LaViola Jr. Symbol recognition dataset. Technical report, Microsoft Center for Research on Pen-Centric Computing.

[20] B. Keshari and S.M. Watt. Online mathematical symbol recognition using svms with features from functional approximation. In *Proc. Mathematical User-Interfaces Workshop 2008, (MathUI 08)*, Birmingham, UK, July 2008.

[21] C. J. Leggetter and P. C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech & Language*, 9(2):171 – 185, 1995.

[22] Zicheng Liu, Henrique S. Malvar, and Zhengyou Zhang. System and method

for ink or handwriting compression. *United States Patent No US 7,302,106 B2*, November 2007.

[23] K. Gulez M. Mercimek and T.V. Mumcu. Real object recognition using moment invariants. *Sadhana*, 37(6):765–775, 2005.

[24] Inc. Maplesoft. Maple 13 user manual. Technical report, Maplesoft, 2009.

[25] V. Mazalov and S.M. Watt. Digital ink compression via functional approximation. In *Proc. 12th International Conference on Frontiers in Handwriting Recognition, (ICFHR 2010)*, Kolkata, India, November 16-18 2010 (accepted).

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

[27] Mario E. Munich and Pietro Perona. Visual signature verification using affine arc-length. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2180, 1999.

[28] O. Golubitsky V. Mazalov and S.M. Watt. Orientation-independent recognition of handwritten characters with integral invariants. In *Proc. Joint Conference of ASCM 2009 and MACIS 2009: Asian Symposium of Computer Mathematics and Mathematical Aspects of Computer and Information Sciences, (ASCM 2009)*, pages 252–261, Fukuoka, Japan, December 2009. COE Lecture Note Vol. 22, Kyushu University, ISSN 1881-4042.

[29] O. Golubitsky V. Mazalov and S.M. Watt. Toward affine recognition of handwritten mathematical characters. In *Proc. International Workshop on Document Analysis Systems, (DAS 2010)*, pages 35–42, Boston, USA, June 9-11 2010. ACM Press.

[30] K. R. Ramakrishnan R. Mukundan. *Moment Functions in Image Analysis: Theory and Applications.* World Scientific, 1998.

[31] I. Kogan S. Feng and H. Krim. Classification of curves in 2d and 3d via affine integral signatures. *to appear in Acta Appl. Math.*, 2008.

[32] Patrice Scattolin. Recognition of handwritten numerals using elastic matching. Master's thesis, Concordia University, Montreal, Quebec, Canada, 1995.

[33] Slate Corporation. *Jot - a specification for an ink storage and interchange format*, May 1996.

[34] Elena Smirnova and Stephen M. Watt. Communicating mathematics via pen-based computer interfaces. In *International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, (SYNASC 2008)*, pages 9–18, Timisoara Romania, September 2008. IEEE Computer Society.

[35] G. Talenti. Recovering a function from a finite number of moments. *Inverse Problems*, (3):501–517, 1987.

[36] V. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[37] Toru Wakahara and Kazumi Odaka. On-line cursive kanji character recognition using stroke-based affine transformation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 19, pages 1381–1385, December 1997.

[38] Toru Wakahara and Seiichi Uchida. Hierarchical decomposition of handwriting deformation vector field using 2dwarping and global/local affine transformation. In *10th International Conference on Document Analysis and Recognition*, pages 1141–1145. IEEE Computer Society, July 2009.

[39] R. Khanna Weicheng Shen. Prolog to face recognition: Eigenface, elastic matching, and neural nets. In *Proceedings of the IEEE*, number 9, pages 1422–1422. IEEE Computer Society, September 1997.

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

Curriculum Vitae

Name	Vadim Mazalov
Place of Birth	Russian Federation
Post-secondary Education	The University of Western Ontario London, Ontario, Canada September 2009 - August 2010 M. Sc.(Computer Science) *Supervisor*: Pr. Stephen M. Watt
	Kuban State University Krasnodar, Russia September 2003 - July 2009 B.S./M.S. (Applied Mathematics and Computer Science)
	Roanoke College Salem, Virginia, USA September 2005 - May 2006 Exchange Student
Related Work Experience	Teaching Assistant The University of Western Ontario September 2009 - August 2010
	Research Assistant Ontario Research Centre for Computer Algebra September 2009 - August 2010
	Software Engineer Peter-Service, CJSC September 2007 - July 2009

Publications

- V. Mazalov and S.M. Watt. Digital Ink Compression via Functional Approximation. *Proc. International Conference on Frontiers in Handwriting Recognition, (ICFHR 2010)*, Kolkata, India (accepted).

- O. Golubitsky, V. Mazalov and S.M. Watt. Toward Affine Recognition of Handwritten Mathematical Characters. In *Proc. International Workshop on Document Analysis Systems, (DAS 2010)*, pp. 35-42, June 9-11 2010, Boston, USA, ACM Press.

- O. Golubitsky, V. Mazalov and S.M. Watt. Orientation-Independent Recognition of Handwritten Characters with Integral Invariants. In *Proc. Joint Conference of ASCM 2009 and MACIS 2009: Asian Symposium of Computer Mathematics and Mathematical Aspects of Computer and Information Sciences, (ASCM 2009)*, pages 252–261, Fukuoka, Japan, December 2009. COE Lecture Note Vol. 22, Kyushu University, ISSN 1881-4042.