Entropy **2012**, *14*, 2324-2350; doi:10.3390/e14112324

OPEN ACCESS

entropy

ISSN 1099-4300

www.mdpi.com/journal/entropy

Article

On Using Entropy for Enhancing Handwriting Preprocessing

Andreas Holzinger [1,*], Christof Stocker [1], Bernhard Peischl [2] and Klaus-Martin Simonic [1]

[1] Research Unit Human-Computer Interaction, Institute for Medical Informatics, Statistics and Documentation, Medical University Graz, Auenbruggerplatz 2/V, A-8036 Graz, Austria; E-Mails: c.stocker@hci4all.at (C.S.); klaus.simonic@medunigraz.at (K.-M.S.)

[2] Softnet Austria, Infeldgasse 16b, A-8010 Graz, Austria; E-Mail: bernhard.peischl@soft-net.at

* Author to whom correspondence should be addressed; E-Mail: a.holzinger@hci4all.at; Tel.: +43-316-385-13883.

Received: 21 July 2012; in revised form: 7 November 2012 / Accepted: 13 November 2012 / Published: 19 November 2012

Abstract: Handwriting is an important modality for Human-Computer Interaction. For medical professionals, handwriting is (still) the preferred natural method of documentation. Handwriting recognition has long been a primary research area in Computer Science. With the tremendous ubiquity of smartphones, along with the renaissance of the stylus, handwriting recognition has become a new impetus. However, recognition rates are still not 100% perfect, and researchers still are constantly improving handwriting algorithms. In this paper we evaluate the performance of entropy based slant- and skew-correction, and compare the results to other methods. We selected 3700 words of 23 writers out of the Unipen-ICROW-03 benchmark set, which we annotated with their associated error angles by hand. Our results show that the entropy-based slant correction method outperforms a window based approach with an average precision of $\pm 6.02^{\circ}$ for the entropy-based method, compared with the $\pm 7.85^{\circ}$ for the alternative. On the other hand, the entropy-based skew correction yields a lower average precision of $\pm 2.86^{\circ}$, compared with the average precision of $\pm 2.13^{\circ}$ for the alternative LSM based approach.

Keywords: entropy; handwriting recognition; point cloud data; preprocessing

---

1. Introduction

Since the introduction of entropy to information theory [1], a lot of research has been concerned with refining and applying the term entropy for different fields of application.

In our work presented in this paper, we study the impact of entropy for the field of handwriting recognition. Why handwriting? Handwriting is more than just the use of pen and paper. It can also be seen as a very natural way of communication between humans and computers. Even though keyboards have proven to be an effective interface, the advancement in computational technology calls for a new or at least different way of human-computer interaction. The ultimate goal here is to make the communication with electronic devices feel as natural as communicating with other humans, without the restriction of an additional learning process. This puts speech recognition and handwriting recognition in the focus of future user interfaces.

In 2007 Steve Jobs argued "Who need a stylus?". Yet interestingly, Apple has made a patent application on stylus [2], and the use of stylus even dates back to the Apple Newton [3]. Undoubtedly, it is true that touch input is well accepted and easy to learn, even amongst non-computer literate people and elderly people [4,5]. Devices with touch screens are useful in hospitals, where patients can, for example, fill out questionnaires while they are waiting for their examination, for the reception by the doctor, or during other spare times [6].

Direct input of questionnaire answers by the patients makes the error prone and time consuming copying of completed paper sheets unnecessary. This saves time, which can be used for direct contact with the patient, thereby improving the overall quality of the interaction between doctors and their patients. Although touch is a very intuitive way of interaction, it was shown that in a professional medical context, styluses are preferred over finger-based input [7].

Input via stylus has the advantage of being more precise and the action is similar to the user's accustomed writing on sheets of paper—and paper is still a preferred medium in the hospital [8]. For addressing the problem of imprecise touch using fingers, Vogel and Baudisch [9] developed a system called Shift, which makes it possible to make more precise selections using a finger. Shift shows a copy of the touched screen location and shows a pointer representing the selection point of the finger if the finger is placed over a small target. However, for further improving the precision of touch input via finger it must be better understood how people touch touch screens [10].

Another problem with touch input using fingers is that the user's "fat fingers" also cover the areas the user intends to touch. To circumvent this problem, [11] developed a mobile device that can be operated from the back. In addition, by using back-of-device interaction, it is possible to create very small touch devices [12].

Despite all these facts, medical professionals (medical doctors, nurses, therapists, first responders, *etc.*) are more familiar with dictation and handling a stylus, since they are used to handling a pen all the time [7,13], despite the issue of poor handwriting in medicine generally [14].

As regards input technology, the most recent development on the mobile market is at contrast to the preferred input technique of professionals in the medical domain, whereas from the viewpoint of Human-Computer Interaction (HCI), handwriting can be seen as a very natural input technology [15]. Studies have shown that a recognition rate below 97% is not acceptable to end users [16]. The challenge

in developing such a system is the fact that the art of handwriting is very individual, making a universal recognition of all handwriting particularly demanding [17].

A typical example is the case of incoming patients in the triage (aka EBA: first clinical examination), where it is similar to an emergency: Rapid patient information collection is crucial. Promptly and accurately recorded and well communicated vital patient data can make the difference between life and death [18,19]. Consequently, the data acquisition should have as little disruptive effect on the workflow of the medical professionals as possible. In the past, solutions for data input on mobile applications have been tested in the field [7,15,17,20–23].

Due to the fact that emergencies are usually complicated by difficult physical situations, special attention has to be given to the design of information technology for emergencies [24]. A key issue of any such information system is the acquisition of textual information. However, extensive text entry on mobile devices is principally to be avoided and a simple and easy to use interface, in accordance with the maxim *less is more*, is a supreme necessity [22].

The basic evidence is that entering data into a mobile device via a stylus is slower, more erroneous and less satisfactory for end users than entering data via a QWERTZ (de) or QWERTY (us) keyboard, as has been demonstrated in some studies [25,26], however, the use of a stylus is much faster and more accurate than using finger touch [7].

We will start by talking about the theory behind handwriting and briefly explain how online handwriting recognition is practised today. After explaining the mathematical background, we will go into related work and talk about how entropy has been used in the field of handwriting for the last 35 years. There is very little application, as of today, in the field of online handwriting recognition. This enabled us to carefully re-implement those processes step by step. After presenting our results, we will conclude this paper by talking about possible future application of entropy in the field of handwriting recognition.

1.1. History of Handwriting Recognition

Handwriting has a long tradition in mankind and goes back to the early cave painters [27]. The ingenious idea of captivating human thoughts into signs and symbols (pictures, later with letters) was a major cultural step. Plato (428–328 BC) described the human memory as a wax tablet and the Romans used a stylus (Latin: stilus) and wax-tablet (Latin: tabula cerata) as handwriting capturing tool, which looks astonishing similar to our stylus and touch-tablets of today (see Figure 1).

With the advent of modern technology, more sophisticated ideas of captivating handwriting emerged: The first patent was issued in the US to Elisa Gray in 1888 for an electrical stylus for captivating handwriting and transmission via telegraph. The first patent on handwriting recognition, as we know it still today, was issued in 1914 and in 1915 to Hyman Eli Goldberg on the on-line recognition of hand-written numerals to control a machine in real-time. In 1945 handwriting recognition was also described by Vannevar Bush within the context of the MEMEX (Memory expander) vision, followed by real-world implementations in form of the Stylator in 1957, the RAND tablet in 1961 and the electronic ink project GRAIL in 1969. For detailed information on the Archaeology of handwriting recognition refer to [28].

Figure 1. A roman writer from Constantinian time (306-337 AD) holding a wax-tablet (Latin: hexaptychon) in his left hand and a stylus in his right hand (Sculpture in the Lapidarium of the Landesmuseum Joanneum in Graz, Eggenberg, picture taken 7 November 2012 by the authors.

1.2. Model of Handwriting

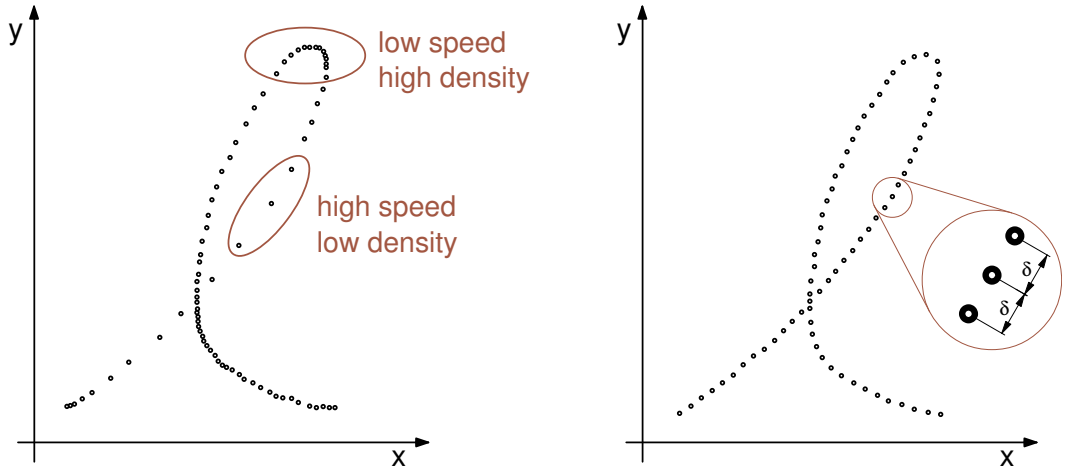
Handwriting recognition (HWR) methods can generally be classified into offline and online recognition. While offline handwriting recognition deals with a bitmap presentation of the handwriting, online handwriting recognition uses the pen trajectory as input. Since in this paper we are concerned with online handwriting recognition, we model the continuous handwriting input signal given by an input device as

$$X(t) = (x(t), y(t), p(t))^T \tag{1}$$

It contains the coordinates $x(t)$ and $y(t)$ as well as the pressure $p(t)$ of the stylus [29]. It might be interesting to note that some devices also provide azimuth and inclination of the stylus [30]. After the digitalization process, $X(t)$ is considered as a discrete time series sampled at different points $t \in T$ over time. Let the sampling times be $t_0, t_1, ..., t_n$, satisfying $0 \leq t_0 < t_1 < ... < t_n$. If the time points are equally spaced (*i.e.*, $|t_{i+1} - t_i| = \tau$ for all $i = 0, 1, ..., n-1$, $\tau > 0$ some constant), we call the input signal *regularly* sampled.

Let $d(X(t_i), X(t_{i+1})) = \left((x(t_{i+1}) - x(t_i))^2 + (y(t_{i+1}) - y(t_i))^2\right)^{1/2}$ be the Euclidian distance with respect to the coordinates $x(t)$ and $y(t)$. A sampling of the handwriting trajectory satisfying $d(X(t_i), X(t_{i+1})) = \delta$, for some constant $\delta > 0$ and $i = 0, 1, ..., m-1$, is referred as the *equidistant* re-sampling of the time series $X(t)$. One notices that $t_m \leq t_n$ holds and in general the equidistant re-sampling is not regular (see Figure 2).

Figure 2. Example of a regularly sampled input signal (left) and its equidistant re-sampling (right).



2. Theoretical Background

Handwriting recognition in general has to overcome a lot of obstacles. To understand handwriting recognition, one has to recognize a few basic things about handwriting in general. Every writer differs in his handwriting-style from one another in a unique way. Sometimes they differ so much that it is even hard for the other person to read it.

Many people do not write in distinct letters with clear spacing between them. Often letters tend to run together, making it harder to separate them. But even if one can segment the characters, identification remains a problem. Some characters, such as I-1, O-0, l-1, 5-S, 6-G tend to look alike, or even the same, again depending on the writer [31]. Hence, sometimes characters are only distinguishable through the context in which they occur [32].

Handwriting recognition is still considered an open research problem, mainly due to the substantial individual variation in appearance. Consequently, the challenges include the distortion of handwritten characters, since different people may use different style of handwriting, direction, *etc.* [33].

If a system needs to deal with the input of different end users, a training phase is required to enable the system to understand the user's art of writing. The data received in this phase is stored in a database. During the recognition process, the system compares the input with the stored data and calculates the output.

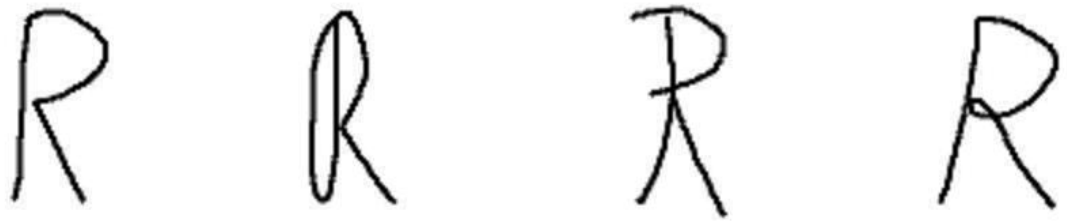
Handwriting can be characterized as a sequence of basic strokes connected according to a rule. Consequently, recognition can be seen as a matching process, used as the fundamental principle in a handwriting recognizer. Early work in handwriting recognition dates back into the 1960s [34].

2.1. Handwriting Properties

Every written language has an alphabet consisting of different characters and most of the time a handful of symbols for punctuation. Handwriting typically consists of different strokes done by the writer. A stroke is the path of the tip of a pen from pen down to pen up. There can be more than one stroke per letter, but there can also be more than one letter per stroke.

A basic principle behind any written language, and the condition that makes communication possible, is that the difference between different characters is greater than the difference between multiple drawings of the same character [31]. An example for different drawings of the letter "R" can be seen in Figure 3. There are however exceptions, as was mentioned before. Some characters, like O and 0, tend to look alike and are sometimes only distinguishable through the context in which they occur [32].

Figure 3. Different drawings of the same character.



In the Latin alphabet, all characters vary in their static and dynamic properties. Static properties would be things like size and shape, while dynamic properties are typically things like stroke number and order [31].

2.2. The Process of Online Handwriting Recognition

The process of online handwriting recognition can be broken down into a few general steps: preprocessing, feature extraction and classification [35].

The purpose of preprocessing is to discard irrelevant information in the input data that can negatively affect the recognition [36]. This means speed and accuracy. Preprocessing usually consists of binarization, normalization, sampling, smoothing and denoising [35].

The second step is feature extraction. Out of the two- or three-dimensional vector field received from the preprocessing algorithms, higher dimensional data is extracted. The purpose of this step is to highlight important information for the recognition model [32]. This data may include information like pen pressure, velocity or the changes of writing direction.

The last big step is classification. In this step various models are used to map the extracted features to different classes and thus identifying the characters or words the features represent.

2.3. Impact of Skew and Slant in Handwriting Recognition

Tang *et al.* [37,38] pointed out the potential of preprocessing by introducing the Entropy-Reduced Transformation, where the goal is to reduce the entropy of the input data set. Since distortions in the input data, like skew or slant, increase the entropy of the data set, they increase the variations within samples that represent the same class and thus making it more challenging for the recognition process to yield good results. While slant correction can often be ignored in writer dependent systems, where the slant might be consistent for the writer, it is especially important for writer independent systems. The

goal is to minimize the variation between different drawings of the same character. Slant correction is also useful to simplify the segmentation procedure.

Brakensiek *et al.* [39] investigated the influence of skew- and slant-correction in the recognition rate, while using entropy-based techniques. In the case of a writer independent system, they reported an increase of 0.9% to a total recognition rate of 86.7% for a test set of 4153 words. They have also shown that the recognition error of their writer independent system can be reduced by about 7% (compared with only re-sampled data). This tells us that good solutions for skew- and slant-correction would in fact be useful.

Kosmala [32] reported a precision of about $\pm 1^\circ$ for an entropy based skew correction approach, with the restriction of enough data points being available. For an entropy based slant correction approach he reported an average precision of $\pm 7.31^\circ$.

Guerfali and Plamondon [40] used the least squares method for skew correction and reported a precision of $\pm 1.3^\circ$ for a range of allowed angles of $\pm 30^\circ$. For their window based slant correction they have shown an average precision of $\pm 6^\circ$, which they considered as acceptable, because the accuracy of the subjects slant was about $\pm 5^\circ$ [40].

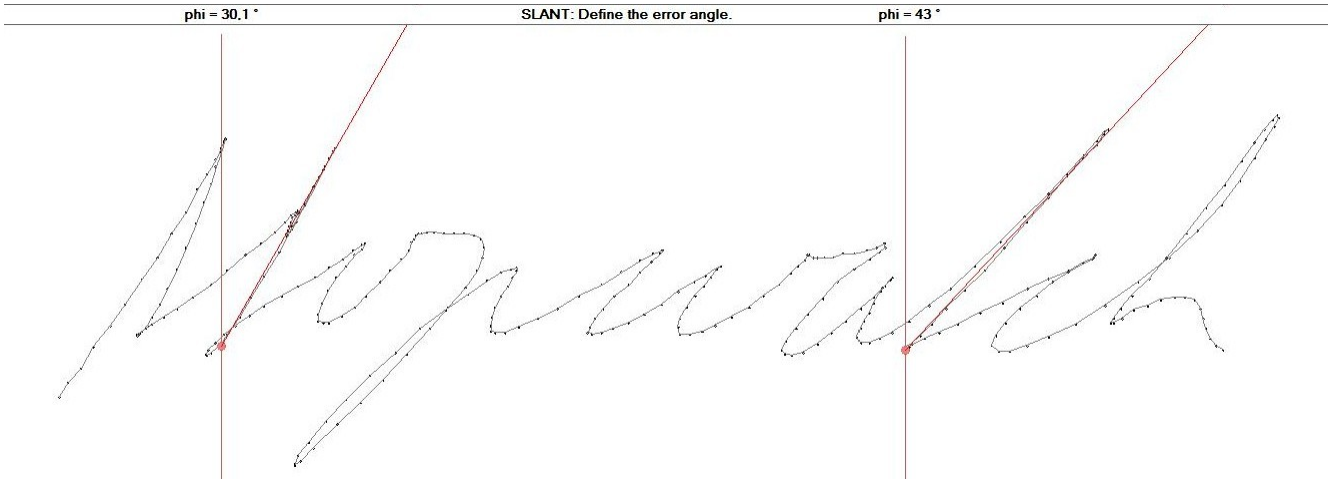
2.4. Challenges of Skew- and Slant-Correction

The effectiveness of skew- and especially slant-correction is still very much dependent on some properties of the input data. Short words, for example, can be a big problem for skew correction. Also, the range of allowed angles can significantly influence the correction precision. A bigger range of angles allow for bigger outliers.

For words written without an available reference line, the base line might not be unique, as the characters tend to vary in size. Also some writers may vary in the shearing angle even within the same word, rendering slant correction with only one angle to describe the distortion impractical. An example for a slant variation of about 13° can be seen in Figure 4. Thus, automatic identification of a unique slant angle might not always be possible—not even with human intervention. To emphasize this, Guerfali and Plamondon [40] asked ten subjects to determine the slant of each one of 275 words. They used the averages of the selected angles as references, while the standard deviations were treated as indication of an acceptable error. Their results have shown that the standard deviation is about $\pm 5^\circ$. This does not necessarily apply for our test results. Nevertheless, it shows that there is in fact an ambiguity involved.

There are promising approaches using local slant. They do so by employing dynamic programming techniques to apply different shearing angles at different points within the word [41]. However, as the algorithm has more freedom to make errors within a word, there are more robustness issues to address [42].

Figure 4. Challenges of slant correction. The angle ϕ varies between 30 and 43 degrees.



2.5. Mathematical Background

In this section we describe the mathematical background necessary to understand the methods we mention in this paper. We start by defining the entropy along with a partitioning. We will use them as correction mechanism for rotation distortions (skew) and sheering distortions (slant).

Let $P = \{p_1, p_2, ..., p_l\}$ be a discrete probability set of a variable X, then the entropy $H(X)$ is defined as

$$H(X) = -\sum_{j=1}^{l} p_j \cdot \log p_j \tag{2}$$

Denote $(\min x(t) = b_0 < b_1 < ... < b_l = \max x(t))$ a partition for an equidistant re-sampling of $x(t)$ with some constant mesh $|\, b_j - b_{j-1}| = w > 0$, $j = 1, 2, ..., l$. The partitioning resembles a binning, which can be used to determine the properties p_j

$$p_j = \frac{1}{m+1} \sum_{t=0}^{m} \chi_j(x(t)) \tag{3}$$

with χ_j being the indicator function for the j-th bin

$$\chi_j(x(t)) = \begin{cases} 1, & b_{j-1} \leq x(t) < b_j \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

As an alternative method for skew correction, we use the least square method to identify the error angle alpha [40]. Let n be the number of minima in X, (x_{t_i}, y_{t_i}) be the coordinates of the minimum point X_{t_i} and t_i be the sampling instant of the i-th minima, then the error angle α is defined as

$$\alpha = \arctan \left(\frac{n \cdot \sum_{i=1}^{n} (t_i \cdot y_{t_i}) - \sum_{i=1}^{n} t_i \cdot \sum_{i=1}^{n} y_{t_i}}{n \cdot \sum_{i=1}^{n} t_i^2 - \left(\sum_{i=1}^{n} t_i \right)^2} \right) \tag{5}$$

Since we deal with distortions, to be precise with rotation and sheering distortions, we need to define correction mechanism once the error angles have been defined.

Let X be the input signal to be rotated by an angle α. Then the skew corrected signal $\overline{X}$ is defined as

$$\overline{X}(t) = R_\alpha \cdot X(t) \tag{6}$$

with R_α being the rotation matrix

$$R_\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{7}$$

Let X be the input signal to be slanted by an angle ϕ. Then the slant corrected signal X' is defined as

$$X'(t) = S_\phi \cdot X(t) \tag{8}$$

with S_ϕ being the sheering matrix

$$S_\phi = \begin{pmatrix} 1 & -\tan\phi & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{9}$$

3. Related Work

Entropy found its way into pattern recognition in a number of ways. In this paper, however, we will focus on handwriting recognition. To be specific, we focus on the online handwriting recognition. However, entropy is also used in optical character recognition. In this section we will briefly describe how entropy is applied in the case of the offline handwriting recognition and the writer identification in online handwriting.

Sesa-Nogueras *et al.* [30] analyzed, from an information theory perspective, the gestures produced by human beings when handwriting a text. They used additional handwriting features, such as azimuth and inclination of the stylus. By analyzing the entropy of on-surface and in-air trajectories, which means that the stylus is either touching the surface or is in the air transitioning between strokes, they showed that the amount of information is similar in both trajectories, which in turn appear to be notably non-redundant.

Handwritten characters vary a lot, which consequently requires measures that reflect the variation for a given set of data. For this purpose, Kim *et al.* [43] defined four properties that a variation measure for character data is supposed to satisfy.

- *Boundedness*: because a variation measure should be *independent* of the size of the image and the number of images in a single data set, *i.e.*, the variation values should be *bounded* by a constant maximum and a constant minimum;
- *Independency*: because a variation measure should be independent of the size of the white area in an image as well as the pen used to create images;
- *Monotonicity*, because a variation measure should increase *monotonically* as the grey area of a data set increases;
- *Constancy*, because a variation measure should be independent of the complexity of the character itself.

The authors demonstrated that none of the variation measures proposed at that time satisfied all four properties. Consequently, they introduced a new variation measure, *Average Entropy Difference*, and showed that it satisfied all four properties.

Average Entropy Difference *Let C be defined as*

$$C = \{(x,y) \mid h(x,y) \neq 0\} \tag{10}$$

$$h(x,y) = -p(x,y)\log_2 p(x,y) - (1-p(x,y))\log_2(1-p(x,y)) \tag{11}$$

where $h(x,y)$ is the entropy of the point (x,y). Denote $|C|$ the cardinality of C, then the Average Entropy Difference V_d is defined as

$$V_d = \begin{cases} \frac{1}{|C|}\sum_{x=1}^{X}\sum_{y=1}^{Y}\nu(x,y), & |C| \neq 0 \\ 0, & |C| = 0 \end{cases} \tag{12}$$

with

$$\nu(x,y) = \frac{h(x,y)}{\alpha \cdot \max\{|h(x+1,y) - h(x,y)|, |h(x,y+1) - h(x,y)|\} + 1} \tag{13}$$

being the Entropy Difference of a point (x,y). Since $\nu(x,y) = 0$ for $(x,y) \notin C$, the Average Entropy Difference is also defined as

$$V_d = \frac{1}{|C|}\sum_{(x,y)\in C}\nu(x,y) \tag{14}$$

Park *et al.* [44] proposed a quality measurement method of gray-scale handwriting data to compare different data objectively. For this they defined an Extended Average Entropy, as an extension of the Average Entropy (AE) in binary-scale. They intended to directly measure the handwriting qualities in a given gray-scale character database. Moreover, they measured the quality of each sample in a class, and classified all data within a class into several groups according to their handwriting qualities. The results of Park *et al.* confirmed that their method was useful for measuring the qualities of handwritten Hangul characters (the Korean alphabet). This Extended Average Entropy (EAE) works as follows:

Extended average entropy *Given M images of size $X \times Y$ with L gray levels $l = \{0, 1, 2, ..., L-1\}$, let the gray level of a pixel (x,y) be denoted by $I(x,y)$. Then, the frequency of the gray level l at position (x,y) is defined as*

$$F(x,y;l) = \sum_{m=1}^{M} C_m(x,y;l) \tag{15}$$

where $C_m(x,y;l) = 1$ when $I(x,y) = l$, i.e.*, the gray level at position (x,y) is equal to l. Therefore, the frequency has a value of $0 \leq F(x,y;l) \leq M$. Furthermore, let $P(x,y;l)$ be probability of the gray level l at position (x,y) and $H(x,y)$ be the corresponding entropy with a logarithmic base of L, then the extended average entropy (EAE) in gray-scale is defined as*

$$E^A = \frac{1}{X \cdot Y}\sum_{x=1}^{X}\sum_{y=1}^{Y} H(x,y) \tag{16}$$

4. Materials and Methods

In this section we will go through preprocessing step by step, while focusing on the normalization techniques using entropy. In particular, we will focus on performing normalization techniques based on the idea of Cote *et al.* [45], as well as Kavallieratou *et al.* [46] and adapted to online handwriting recognition by Kosmala [32] as well as Schenk and Rigoll [29].

We will also describe alternative methods for the normalization processes, like skew correction with the use of the least squares method described by Guerfali and Plamondon [40], which we will later use to benchmark the entropy based methods against.

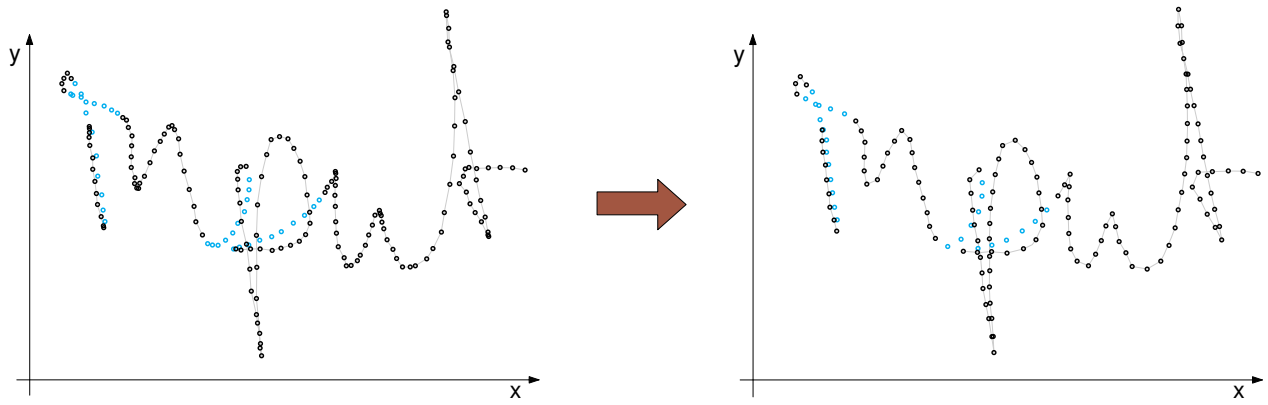
4.1. Sampling

Most of the time sampling, sometimes referred to as filtering [40], is performed after normalization and may as well be the last step of preprocessing. However, it is also common to start the preprocessing with sampling as well, since it is a low cost process that can significantly increase the performance and accuracy of the other preprocessing steps, especially for normalization techniques that use the projection profile of the input data along an axis, as we will explain further down.

There are usually two reasons to perform sampling. One is to remove unnecessary detail in the form of over-fitting (too many points representing the drawing). The other one is to remove artifacts like pen velocity out of the input data. Keep in mind that the pen velocity could still be interesting for the recognition [47]. In that case it has to be extracted and stored before applying the sampling process.

As mentioned before, the continuous input signal $X(t)$ is in general regularly sampled, resulting in an equidistant time series data concerning the time dimension [32]. In some cases, such as in on-line handwritten whiteboard note recognition, the digitalized data may neither be equidistant in time nor in space [29]. However, the goal of sampling remains the same, namely to re-sample the data to be equidistant in space, ideally without distorting the data. An example for data before and after the sampling process can be seen in Figure 5.

Figure 5. Visualization of the re-sampling process. The regularly sampled data (left) and its equidistant re-sampling (right).



4.2. Normalization

The normalization process is concerned with removing the variation of size and position information out of the input data. Most of the time this step consists of scaling and translating, in order to define a common base for recognition and give the writer the freedom to define the size he or she wants to write in. However, depending on the freedom the writer may have, it may also include correction mechanism for skew and slant. Again, the purpose of normalization is to define a common base, so that the recognizer does not have to deal with different sizes, skews and so forth.

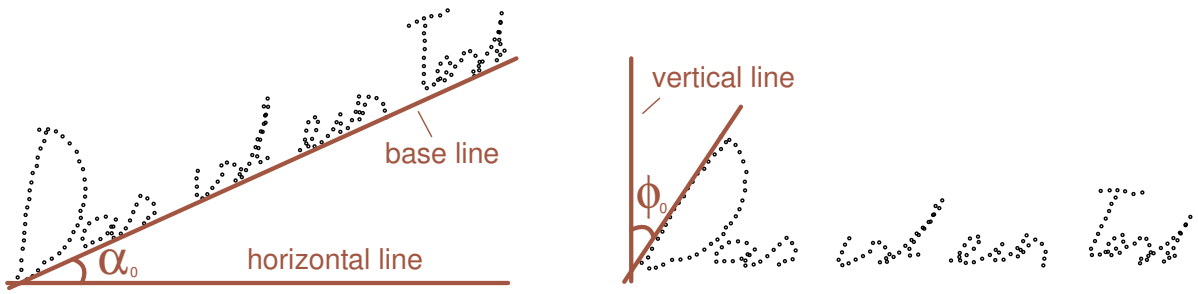
4.2.1. Skew Correction Based on Entropy

Skew correction, or sometimes referred to as baseline drift correction [40], is aimed at bringing the writing direction to the horizontal level.

A typical user interface for handwriting today contains some sort of indication on where to draw your character, word or sentence [48,49]. This may be in the form of a line or even a box. Hence, usually the drawing has the right orientation and does not have to be rotated. But if this is not the case, and the user is presented with the freedom to choose the user's own point of reference, the input data might have a skew that needs to be corrected.

If skew correction needs to be done, the first issue that has to be identified is the error angle α_0. This is the angle between the base line and the horizontal line, as can be seen on the left side in Figure 6.

Figure 6. Visualization for the error angles α_0 and ϕ_0 for performing skew- (left) or slant- (right) correction. The right figure is already skew corrected.



Once the error angle α_0 has been identified, the data points have to be rotated. With respect to Equation (6), let $X(t)$ be the time series after an equidistant re-sampling then the corresponding time series after the rotation is defined as

$$X_\alpha(t) = R_\alpha \cdot X(t) \tag{17}$$

A promising approach described by Kosmala [32] to identify the error angle α_0 is to calculate the projection profile histogram $p_{y,j}(\alpha)$ for a range of binns $1 \leq j \leq l$. The minimum of the entropy distribution of $p_{y,j}(\alpha)$ for a range of angles α is then calculated.

We state the following assumption:

Hypothesis 1 *The slope angle of the baseline, of an equidistant re-sampled time series representing a Latin based word, is closest to* $0\,^\circ$ *when the most* y*-coordinates of the time series are concentrated within some interval* $[y_a, y_b]$, $|y_a - y_b| \leq h_{core}$, *where* h_{core} *is the height of the lower case letters.*

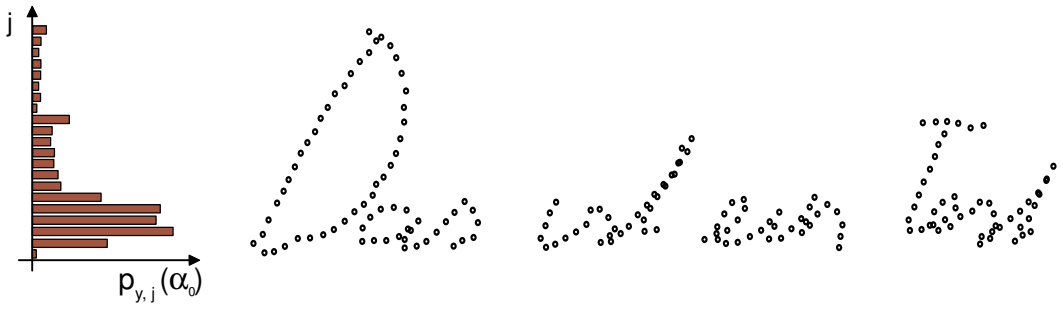
Note that the height of the lower case letters is unknown. However, the entropy of a projection profile histogram gives a good indication on degree of order in the distribution. Hence if most of the data points are located within a small interval, the entropy is supposedly at its minimum.

We define projection profile histogram it in respect to Equation (3) for the y- axis and a range of different angles α as

$$p_{y,j}(\alpha) = \frac{1}{m+1} \sum_{t=0}^{m} \chi_j(y_\alpha(t)) \tag{18}$$

with $y_\alpha(t)$ being the y-component of $X_\alpha(t)$. An example for such a histogram can be seen in Figure 7.

Figure 7. Skew correction of a equidistant re-sampling of the handwriting trajectory $X(t)$. It was performed with the help of the projection profile histograms $p_{y,j}(\alpha)$. This image shows the histogram for the corrected angle of $\alpha_0 = 22°$.

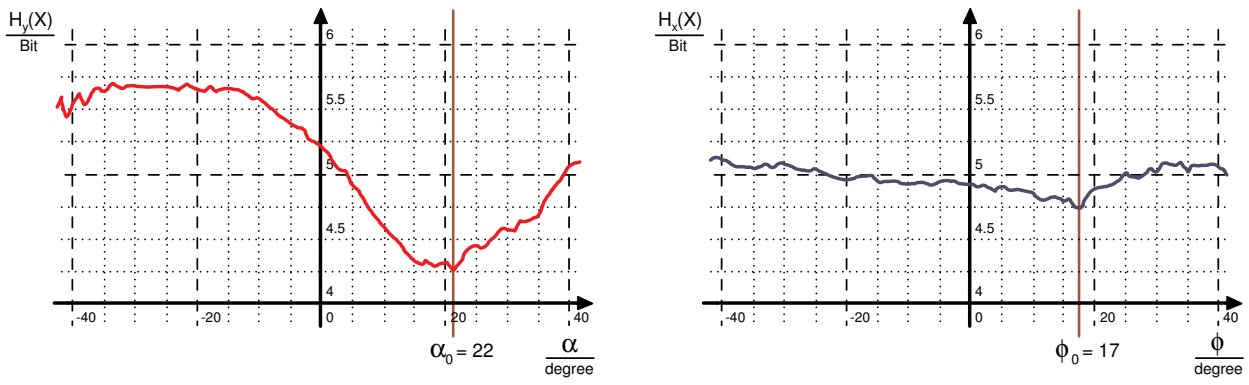


After calculating the projection profile $p_{y,j}(\alpha)$, the corresponding entropy $H_{y,\alpha}(X)$ is computed based on Equation (2) as

$$H_{y,\alpha}(X) = -\sum_{j=1}^{l} p_{y,j}(\alpha) \cdot \log_2 p_{y,j}(\alpha) \tag{19}$$

α_0 is set to the angle α where $H_{y,\alpha}(X)$ is at its minimum, as can be seen in Figure 8.

Figure 8. Skew- and slant-correction performed with the entropy of various projection profile histograms [29,32]. These images show the entropy distributions for different angles. The error angles α_0 and ϕ_0 are set to the angle where the corresponding entropy distribution is at its minimum. Note that the entropy distribution of $H_{x,\phi}$ (right side) is in general not as distinctive as the one of $H_{y,\alpha}$ (left side).



This approach for skew correction supposedly can yield a high precision of about $\pm 1\,^{\circ}$, if there are enough data points available [32].

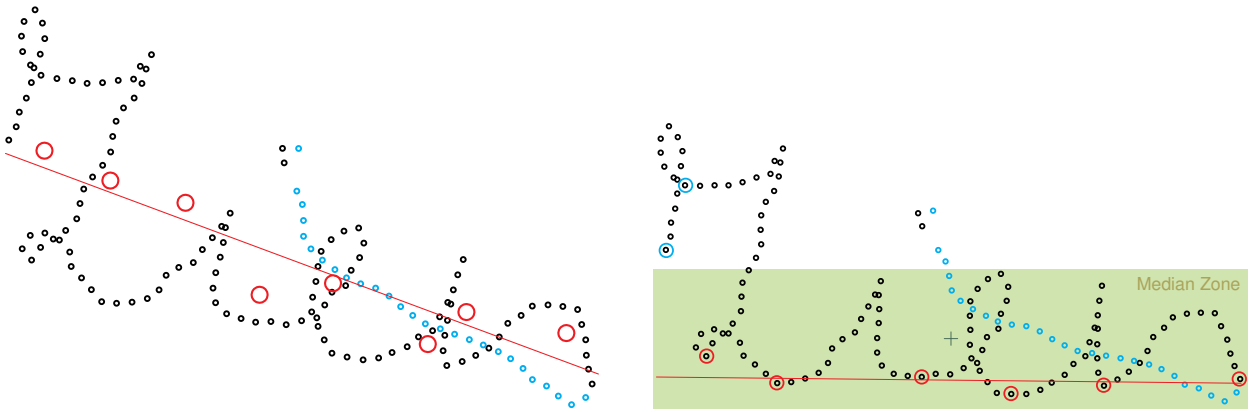
4.2.2. Skew Correction Based on Least Squares Method (LSM)

An alternative approach for skew correction based on the least squares method was introduced by Guerfali and Plamondon [40]. The idea is that the writing baseline is defined as the best fitting straight line passing through the minima of the character drawings [40].

Their approach is divided into two steps. The first step is aimed at detecting major deviations, which can cause errors in locating the real minima of the character drawings. The whole drawing is divided into eight equally spaced regions. For each region, the center of mass of all points within the region is calculated. The eight resulting points serve as input for the LSM to calculate an approximation for the real baseline. The angle α is set to the angle between the calculated baseline and the horizontal.

The second step uses a retroactive process consisting of successive estimations and rotations until satisfactory results are achieved [40]. For this purpose the current minima of the drawing are located. To eliminate superfluous points, only minima within the median zone are used as input for the LSM. Again, the angle α is set to the angle between the calculated baseline and the horizontal. The second step is repeated until an estimated baseline angle of less than $\pm 2\,^{\circ}$ is reached [40]. Both steps are visualized in Figure 9.

Figure 9. Skew correction performed with the least squares method [40]. Step 1 uses the centers of mass of eight regions to compute the regression line can be seen on the left. Step 2 uses the minima within the median zone for the regression line on the right.



4.2.3. Slant Correction Based on Entropy

Slant correction is a little more problematic than skew correction. The slant is described by the error angle ϕ_0, which is the angle between the vertical line and those lines of the drawn text that are supposed to be vertical. Also, the slant may vary along the drawing and might not be the same for all characters as was mentioned in Section 2.4 and can be seen in Figure 4.

The error angle ϕ_0 describing the slant can be computed similar to the error angle α_0 mentioned in entropy based skew correction. The difference is that instead of calculating the relative occurrence of the

data points along the y-axis, the relative occurrence along the x-axis $p_{x,j}(\phi)$ is used. The idea is that the entropy of the relative occurrence is at its minimum when all characters are oriented straight up [29,32]. However, the minimum of the entropy distribution might not always be unique, or correct for that matter. In a lot of cases, slant correction is not deemed to be applicable enough just now [32].

Once the error angle ϕ_0 has been identified, the slant correction can be performed similar to the way the skew correction was performed. In respect to Equation (8), let $X_\alpha(t)$ be the time series after skew correction then the corresponding time series after slant correction is defined as.

$$X_\phi(t) = S_\phi \cdot X_\alpha(t) \tag{20}$$

For slant correction, we define the projection profile histogram with respect to Equation (3), but this time for the x-axis and a range of different angles ϕ

$$p_{x,j}(\phi) = \frac{1}{m+1} \sum_{t=0}^{m} \chi_j(x_\phi(t)) \tag{21}$$

with $x_\phi(t)$ being the x-component of $X_\phi(t)$.

After calculating the projection profile histogram $p_{x,j}(\phi)$, the corresponding entropy $H_{x,\phi}$ is computed similar to skew correction and based on Equation (2).

$$H_{x,\phi}(X) = -\sum_{j=1}^{l} p_{x,j}(\phi) \cdot \log_2 p_{x,j}(\phi) \tag{22}$$

ϕ_0 is set to the angle ϕ where $H_{x,\phi}(X)$ is at its minimum, as can be seen in Figure 8.

4.2.4. Script Line Identification

Script line identification is an important step in the normalization process. The primary purpose of the script lines in preprocessing is size normalization. However, they may also serve for feature extraction [47]. In fields like whiteboard note recognition, where the script lines are much harder to detect, the script line identification process may also serve as a form of de-noising, in the sense that the data points might be modified in the process [47].

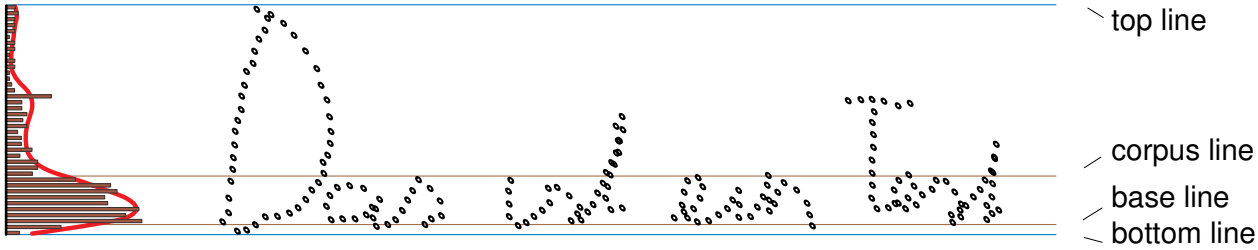
There are four lines that have to be identified. Top line and bottom line are where the maximum and minimum y coordinates of the data points are, and the whole drawing is between top and bottom line. However, those two lines alone are not enough information for scaling, because ascenders and descenders may distort the font size information of the drawing. Good indications for the font size would be the height of the lower case letters in the drawing. For that purpose, corpus line and base line have to be identified. An example for script lines can be seen in Figure 10.

One approach to compute corpus- and base line is to use the relative occurrence $p_{y,j}$ of the input data as it was described in skew correction and compute (or in the concrete case approximate) the first derivative [29]

$$p'_{y,j} = \frac{\mathrm{d}}{\mathrm{d}j} p_{y,j} \tag{23}$$

The minimum and maximum of the derivative are a good indication for the script line positions in general. Of course, the accuracy depends a lot on the proper alignment of the text [29]. A visualization of this approach can be seen in Figure 10.

Figure 10. Visualization for script line identification utilizing the occurrence histogram along the y-axis. Note that in this example the corpus- and the base line may not be optimal. The reason for that are variations in size and position of the text, probably caused by the lack of reference lines in the writing process.



Another way to compute the corpus- and the base line would be by putting regression lines through the upper and lower turning points of the data itself [32], or a combination of both approaches [40].

4.2.5. Slant Correction Based on Window

Another approach for slant correction was proposed by Guerfali and Plamondon [40] and is based on an offline slant correction method introduced by Bozinovic and Srihari [50].

First, the script line identification (See Section 4.2.4) is performed in order to identify three possible regions: the upper zone between top line and corpus line (ascenders), the lower zone between base line and bottom line (descenders) and the central region of the middle zone, which is between corpus line and base line. Within those three regions, observation windows are extracted. Each one of those windows is then divided into an upper part and a lower part [40]. For every existing part, the center of mass of all data points within the part is computed. The local slant of a window is then defined as the slope angle of the line connection the mass centers of the upper and lower part of the window. The error angle ϕ_0 of the word is then set to the average of the local slants.

5. Results and Discussion

In this section we will describe our implementation of the previously mentioned methods, as well as discuss the results we obtained in the process. Where possible, we use the same notation and variables for the pseudo-code algorithms, as we did in Section 4.

5.1. Implementation

We start our preprocessing with sampling, as it is described in Section 4.1. For the interpolation of the points, we used the Euclidean distance, which is a straight forward approach. Given enough data points to start with, there is no visible distortion of the input data.

The next step is skew correction. We implemented the entropy-based approach as well as the one based on the least squares method. Both methods were described earlier.

For the entropy based approach we decided to copy the input data set and rotate the copy for a range of different angles. The algorithm for rotation is described in Algorithm 1.

Algorithm 1 Rotating the data points in X for α degree

Require: $K =$ number of data points in X

function ROTATEDATAPOINTS(X, K, α)
  $X_\alpha :=$ new vector of size K
  for $k = 1 \rightarrow K$ **do**
    $X_\alpha[k].x = X[k].x \cdot \text{COS}(\alpha) - X[k].y \cdot \text{SIN}(\alpha)$
    $X_\alpha[k].y = X[k].x \cdot \text{SIN}(\alpha) + X[k].y \cdot \text{COS}(\alpha)$
    $X_\alpha[k].p = X[k].p$ ▷ Optional: pen pressure
  end for
  return X_α
end function

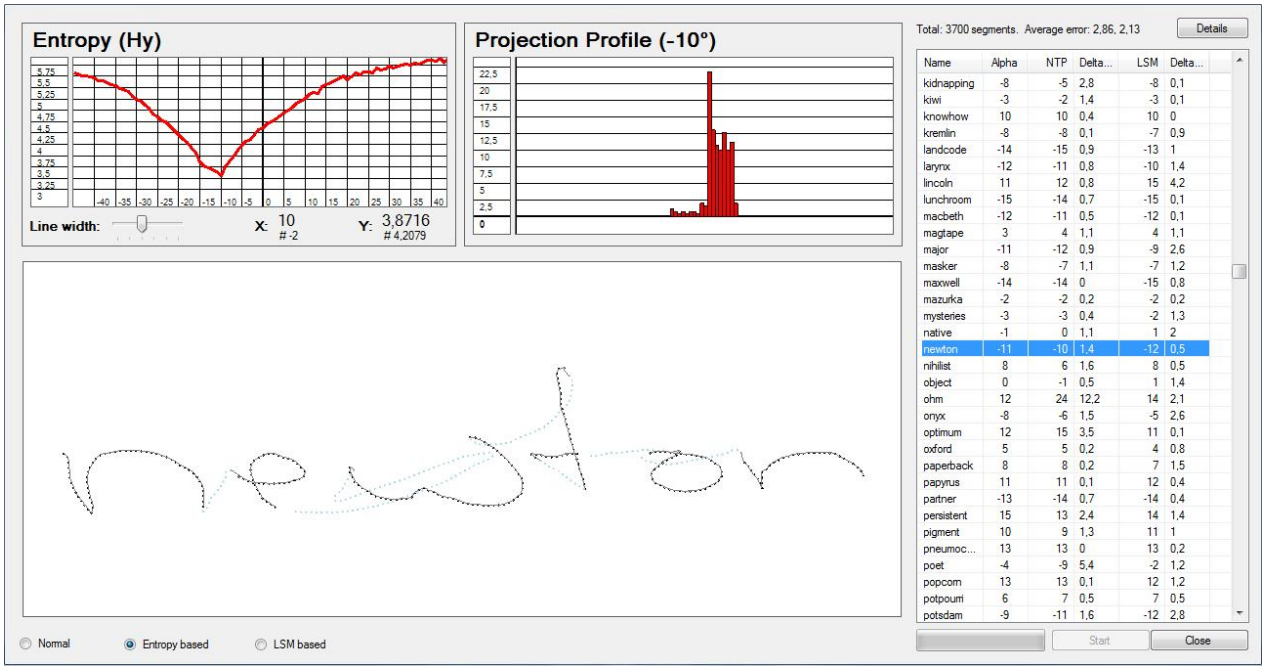
With rotateDataPoints" defined we can calculate the projection profile $p_{y,j}(\alpha)$ for a range of different angles and with those we can compute the entropy $H_{y,\alpha}(X)$ for each angle. We implemented the algorithm as it is described in Algorithm 2. A screenshot of data after the entropy based skew correction process can be seen in Figure 11.

Algorithm 2 Calculate the entropy $H_{y,\alpha}(X)$ for the projection profiles $p_{y,j}(\alpha)$ for a range of angles α

Require: $K =$ number of data points in X
Require: $\forall k \in \mathbb{N}, 1 \leq k \leq K$: $y_{\min} \leq X[k].y$ **AND** $y_{\max} \geq X[k].y$
Require: $l, \alpha_{\min}, \alpha_{\max} \in \mathbb{N}$ **AND** $-35 \leq \alpha_{\min} < \alpha_{\max} \leq 35$

function CALCULATEHY($X, K, \alpha_{\min}, \alpha_{\max}, y_{\min}, y_{\max}, l$)
  $range \leftarrow |\alpha_{\max} - \alpha_{\min}|$
  $H_y :=$ new vector of size $range$ ▷ Denote the index range from $\alpha_{\min}$ to $\alpha_{\max}$
  for $\alpha = \alpha_{\min} \rightarrow \alpha_{\max}$ **do**
    $X_\alpha \leftarrow$ ROTATEDATAPOINTS(X, K, α)
    $w \leftarrow |y_{\max} - y_{\min}| \ / \ l$
    $p_y \leftarrow$ CALCULATECURRENTPY($X_\alpha, K, y_{\min}, y_{\max}, w$)
    $H_y[\alpha] \leftarrow 0$
    for $j = 1 \rightarrow l$ **do**
      $H_y[\alpha] \leftarrow H_y[\alpha] + p_y[j] \cdot \log_2 p_y[j]$
    end for
    $H_y[\alpha] \leftarrow -1 \cdot H_y[\alpha]$
  end for
  return H_y
end function

Figure 11. Screenshot of some example data after skew correction.



The function calculateHy" uses a yet undefined function called calculateCurrentPy". What calculateCurrentPy" does is calculating the projection profile $p_{y,j}(\alpha)$ of the data points in X_α. This means that for every angle, the input data is copied and rotated, in order to calculate the projection profile for that angle. The implementation is described in detail in Algorithm 3.

Algorithm 3 Projection profile $p_{y,j}(\alpha)$ for the current data points in X_α

Require: $K =$ number of data points in X

Require: $\forall k \in \mathbb{N}, 1 \leq k \leq K$: $\quad y_{\min} \leq X_\alpha[k].y \quad$ **AND** $\quad y_{\max} \geq X_\alpha[k].y$

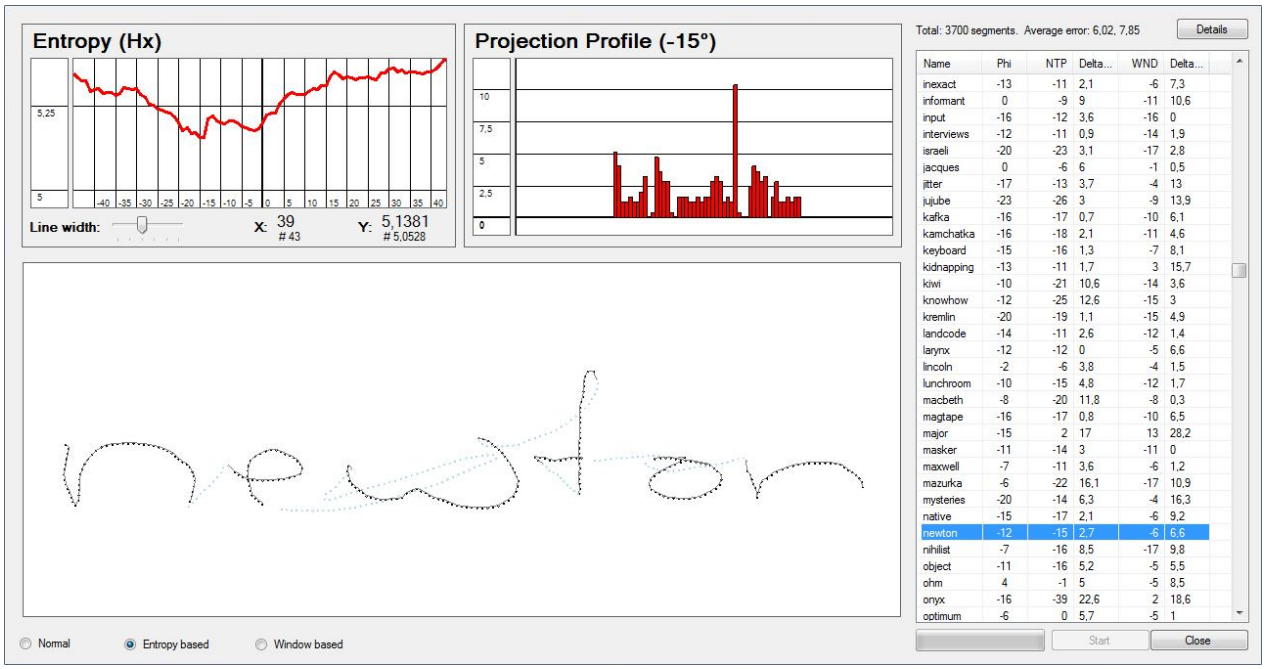
Require: $w < |y_{\max} - y_{\min}| \quad$ **AND** $\quad |y_{\max} - y_{\min}| \mod w = 0$

Ensure: $\sum_j p_y[j] = 1 \pm \varepsilon, \quad \varepsilon \ll 10^{-3}$ ▷ Rounding error depends on data type

function CALCULATECURRENTPY($X_\alpha, K, y_{\min}, y_{\max}, w$)
  $l \leftarrow |y_{\max} - y_{\min}| \,/\, w$
  $p_y :=$ new vector of size l
  for $j = 1 \rightarrow l$ **do**
    $p_y[j] \leftarrow 0$
  end for
  for $k = 1 \rightarrow K$ **do**
    $y_{\text{offset}} \leftarrow X_\alpha[k].y - y_{\min}$
    $j \leftarrow \lfloor y_{\text{offset}} \,/\, w \rfloor + 1$
    $p_y[j] \leftarrow p_y[j] + 1 \,/\, K$
  end for
  return p_y
end function

After skew correction, we apply slant correction to the data set. As we described in Section 4, the algorithms for slant correction are very similar to those of skew correction. The major differences are the axis and the distortion function (sheer instead of rotate). A screenshot of data after the entropy based slant correction process can be seen in Figure 12. The implementation of the sheering function can be seen in Algorithm 4.

Figure 12. Screenshot of some example data after slant correction.



Algorithm 4 Slant the data points in X for ϕ degree

Require: K = number of data points in X

function SLANTDATAPOINTS(X, K, ϕ)
  $X_\phi :=$ new vector of size K
  for $k = 1 \rightarrow K$ **do**
    $X_\phi[k].x = X[k].x - X[k].y \cdot \text{TAN}(\phi)$
    $X_\phi[k].y = X[k].y$
    $X_\phi[k].p = X[k].p$ ▷ Optional: pen pressure
  end for
  return X_ϕ
end function

5.2. Results

We implemented a test suite to compare the effectiveness of the different methods we described. For the test set, we carefully selected a subset of the freely available *The Unipen-ICROW-03 benchmark set* by the *International Unipen Foundation* [51]. To be precise, we selected 3700 words of 23 writers,

consisting out of 737 short words (4 or less letters), 1758 medium words (5 to 7 letters) and 1205 long words (more than 7 letters). Each word is labeled with its length in letters and was measured by a human to define the "ideal" skew- and slant-angles. Consequently, the error angle is the difference between the measured and the computed angle. However, the data set offers a lot of variety in writing styles, rendering angle measuring ambiguous in some cases.

5.2.1. Skew Correction

For skew correction, both approaches show similar properties. Their accuracy is influenced a lot by the word length as can be seen in Figure 13 on the right. In the case of the entropy based method, length and height of the word might get confused when the word is short enough. In the case of the LSM based approach, it is the writing direction that often gets wrongly estimated for short words. Both methods offer an average precision of less than $\pm 1^\circ$ for words with more than 8 letters, as can be seen in Figure 13 on the left. For more detailed results, see Table 1.

Figure 13. Results of skew correction. Average error angle (degree) depends on the length of the words (number of letters). Note that although not significantly, the least squares method approach (blue) outperforms the entropy-based (red) one. Both methods have difficulties dealing with short words.

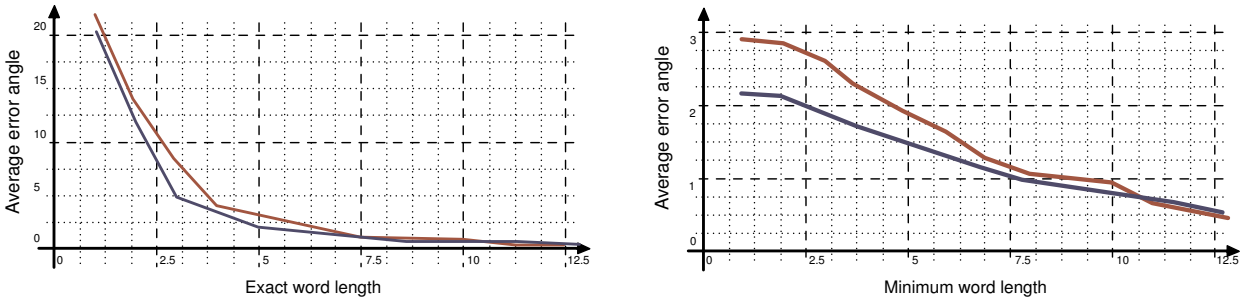


Table 1. Results of skew correction in degree. "Exact word length" means that only words with exactly the given number of letters are considered, while "minimum word length" means that all words that have the given or a greater number of letters are considered.

Exact word length	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Entropy-based method	21.8	14.9	8.1	4.1	2.9	2.5	2.0	1.3	1.2	1.1
LSM-based method	20.3	13.1	5.0	3.7	2.3	2.1	1.8	1.2	1.1	1.1
Minimum word length	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Entropy based method	2.9	2.8	2.6	2.2	1.9	1.7	1.3	1.1	1.0	1.0
LSM based method	2.1	2.1	1.9	1.7	1.5	1.3	1.2	1.0	0.9	0.8

For the whole test set, the entropy based approach yields an average precision of $\pm 2.86^\circ$, while the LSM based approach yields an average precision of $\pm 2.13^\circ$.

5.2.2. Slant Correction

In our case, the entropy-based slant correction yields a little better result than the originally reported result by Kosmala [32]. Similar to skew correction, both mechanisms have problems with very short words and improve as the word length increases. However, as can be observed in Figure 14, the average error starts to increase again when words are getting longer than 6–7 letters. We think that a non-uniform slant is the reason for this. The longer a word gets, the more the slant tends to vary. Also our test set does not include a high percentage of words with more than 10 letters (less than 150 words). This might result in a distortion of the average error distribution. For detailed results have a look at Table 2.

Figure 14. Results of slant correction. Average error angle (degree) depends on the length of the words (number of letters). The entropy-based approach (red) outperforms the window-based approach (blue) for our test set.

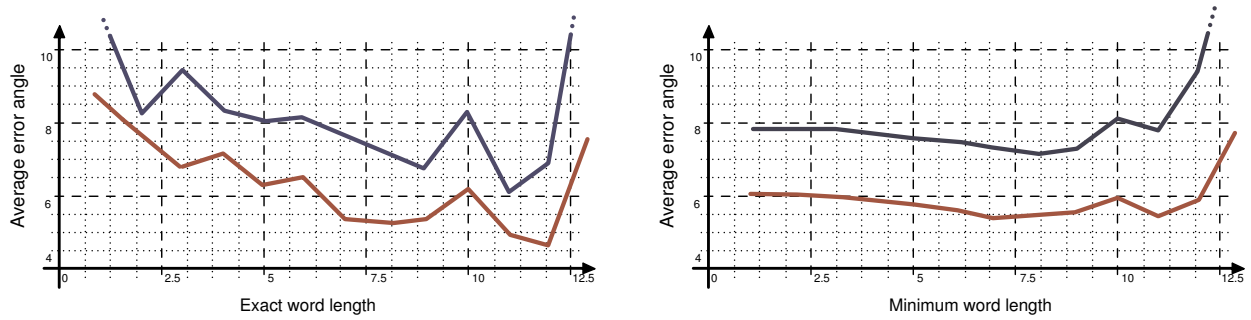


Table 2. Results of slant correction in degree. "Exact word length" means that only words with exactly the given number of letters are considered, while "minimum word length" means that all words that have the given or a greater number of letters are considered.

Exact word length	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Entropy-based method	8.8	7.7	6.8	7.1	6.3	6.5	5.4	5.3	5.5	6.2
LSM-based method	10.9	8.3	9.5	8.3	8.0	8.1	7.7	7.1	6.7	8.3
Minimum word length	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Entropy-based method	6.0	6.0	6.0	5.9	5.7	5.6	5.5	5.5	5.5	6.0
LSM-based method	7.9	7.9	7.9	7.8	7.5	7.5	7.4	7.1	7.4	8.1

For the whole test set, the entropy-based approach yields an average precision of $\pm 6.02°$, while the window-based approach yields an average precision of $\pm 7.85°$.

6. Future Work

Although soft virtual keyboards are well-established in the B2C market, e.g., iPad and tablet computers running on Android, handwriting offers many benefits in contrast to a keyboard, particularly if it comes to using mobile devices in B2B applications. The tremendous success of tablet computers

suggests bringing the underlying concepts (User Interface design, social media, mobile devices) into the B2B market and enriching the proven UI concepts with a stylus-based handwriting recognition.

Briefly we can summarize the benefits of handwriting as follows. Firstly, it feels more natural as we have learned writing with pen and paper very early at school. Secondly, using a pen instead of a keyboard has the advantage that one can immediately sketch and draw. This process supports communication and enhances creativity in a natural way. Thirdly, the ongoing trend of "bringing your own device" (ByoD [52]) in many businesses allows for enriching the positive user experience on the own device in terms of a handwriting application used in the professional (B2B) field: Employees are familiar with handling their individual device and could use a stylus-based handwriting application where suitable for the specific task in mind. Moreover, companies amongst the various sectors start introducing tablet computers in their daily working process [53]. As the daily use of tablets is a precondition for deploying handwriting applications, it is important to overcome the following obstacles:

- Tablet computers have originally been developed for private use only. It is thus important to ensure appropriate policies for the security of sensitive data in the B2B domain.
- IT departments have to ensure appropriate access policies as they have to know who is allowed to access which kind of data pools in the enterprise. This raises challenges in the field of identity management, as employees access the relevant data pools from various locations with different devices.
- Configuration and software deployment usually need to be handled using a central authority. This includes establishing inventories and automatic software updates and security measures.
- In some cases establishing an own tablet infrastructure has proven successful (also see [53]).

Once these obstacles have been mastered, the companies can benefit from introducing mobile devices like tablets the daily working process. As an example, we briefly mention two application scenarios in which using a stylus might offer benefits. In both cases, requirements from the core business suggest the usage of handwriting recognition. The first example stems from the field of medicine. Previous research has shown that medical doctors are accustomed to using a pen in their daily work and have a high willingness to use an electronic stylus as input device [7], whereas elderly people or people who are not computer literate prefer their finger as input medium [5]. Further, as medical doctors usually wear gloves, many of the UI concepts (multi-touch, capacitive displays, *etc.*) do not work in this field. The medical domain is thus a perfect test-bed for introducing handwriting recognition.

The second example is from the insurance industry. With the consequences of the financial crisis starting in 2008, the requirements for insurance companies and financial brokers have become much stricter. According to the MIFID directive (Markets in Financial Instruments Directive [54]), an insurance company (or the broker selling insurance products) is required to document the process of consultancy in detail, including the consideration of the risk awareness of the customer. The fact that the sales and consulting staff of insurance companies typically act at the customer site suggests equipping these departments with mobile devices. In this way, any kind of relevant information regarding new products (e.g., Produpedia, also see [55]) and the documentation of the sales and consulting process can be deployed on a tablet and any kind of (handwritten) documentation can be immediately processed

after entering the relevant information at the site of the customer. In this respect, a working solution for handwriting recognition is a valuable contribution to deal with these directives. In particular, if approval by signature is part of the process, a handwriting solution is desirable since a stylus is anyway in place in this case. Popular examples of using tablets (alongside with social media and the idea to bring the successful concepts from the B2C market to the B2B market) include Deutsche Bank and SAP [53].

7. Conclusions

In this paper we demonstrated the strengths and weaknesses of using entropy for skew- and slant-correction. We showed that for our test set, the entropy based skew correction does not outperform older methods like skew correction based on the least squares method. In fact it is the other way around. We suspect that the noise in the drawing distorts the real minima of the entropy distribution. In many cases where the global minimum was the wrong choice, there was a local minimum close to the real error angle. Even though both approaches yield satisfying result for words longer than five letters, we suggest further investigation into the entropy-based skew correction method, with noise reduction in mind.

On the other hand, we demonstrated that entropy is in fact useful when performing slant correction, as it does outperform the window-based approach. We have come to the conclusion that the window-based method is too much dependent on a number of factors. Its performance is influenced a lot by the outcome of zone detection and by the writing style of the writer. It is also influenced a lot by window selection. We think that the entropy-based slant correction is more stable in that regard.

We also pointed out that all four preprocessing techniques have problems when it comes to short words or letters. In real world applications, it might be useful to identify or approximate the word length before applying any of the above described techniques.

Acknowledgments

The work presented herein has been partially carried out within the competence network Softnet Austria II ([56], COMET K-Projekt) and funded by the Austrian Federal Ministry of Economy, Family and Youth (bmwfj), the province of Styria, the Steirische Wirtschaftsfoerderungsgesellschaft mbH (SFG), and the city of Vienna in support of the Center for Innovation and Technology (ZIT).

References

1. Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423.
2. Yaeger, L.S.; Fabrick, R.W.; Pagallo, G.M. Patent on a Method and Apparatus for Acquiring and Organizing Ink Information in Pen-Aware Computer Systems. IPO 20090279783, 12 November 2009.
3. Yaeger, L.S.; Webb, B.J.; Lyon, R.F. Combining neural networks and context-driven search for on-line, printed handwriting recognition in the newton. In *Neural Networks: Tricks of the Trade*; Springer-Verlag: London, UK, 1998; pp. 275–298.

4. Holzinger, A. User-centered interface design for disabled and elderly people: First experiences with designing a Patient Communication System (PACOSY). In *Computers Helping People with Special Needs*; Miesenberger, K., Klaus, J., Zagler, W., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2398, pp. 467–484.
5. Holzinger, A. Finger instead of mouse: Touch screens as a means of enhancing universal access. In *Universal Access: Theoretical Perspectives, Practice, and Experience*; Lecture Notes in Computer Science; Springer-Verlag: Berlin/Heidelberg, Germany, 2003; pp. 387–397.
6. Holzinger, A.; Kosec, P.; Schwantzer, G.; Debevc, M.; Hofmann-Wellenhof, R.; Frühauf, J. Design and development of a mobile computer application to reengineer workflows in the hospital and the methodology to evaluate its effectiveness. *J. Biomed. Inform.* **2011**, *44*, 968–77.
7. Holzinger, A.; Höller, M.; Schedlbauer, M.; Urlesberger, B. An investigation of finger versus stylus input in medical scenarios. In Proceedings of the 30th International Conference on Information Technology Interfaces, Lisbon, Portugal, 23–26 June 2008; pp. 433–438.
8. Holzinger, A.; Baernthaler, M.; Pammer, W.; Katz, H.; Bjelic-Radisic, V.; Ziefle, M. Investigating paper *vs.* screen in real-life hospital workflows: Performance contradicts perceived superiority of paper in the user experience. *Int. J. Hum.-Comput. Stud.* **2011**, *69*, 563–570.
9. Vogel, D.; Baudisch, P. Shift: A technique for operating pen-based interfaces using touch. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, San Jose, California, USA, 30 April–3 May 2007; ACM: New York, NY, USA, 2007; pp. 657–666.
10. Holz, C.; Baudisch, P. Understanding touch. In Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems, Vancouver, BC, Canada, 7–12 May 2011; ACM: New York, NY, USA, 2011; pp. 2501–2510.
11. Wigdor, D.; Forlines, C.; Baudisch, P.; Barnwell, J.; Shen, C. Lucid touch: A see-through mobile device. In Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology, Newport, RI, USA, 7–10 October 2007; ACM: New York, NY, USA, 2007; pp. 269–278.
12. Baudisch, P.; Chu, G. Back-of-device interaction allows creating very small touch devices. In Proceedings of the 27th International Conference on Human Factors in Computing Systems, Boston, MA, USA, 4–9 April 2009; ACM: New York, NY, USA, 2009; pp. 1923–1932.
13. Holzinger, A.; Höller, M.; Bloice, M.; Urlesberger, B. Typical problems with developing mobile applications for health care—Some lessons learned from developing user-centered mobile applications in a hospital environment. In Proceedings of the International Conference on e-Business, Porto, Portugal, 26–29 July 2008; Filipe, J., Marca, D.A., Shishkov, B., van Sinderen, M., Eds.; INSTICC Press: Porto, Portugal, 2008; pp. 235–240.
14. Sokol, D.K.; Hettige, S. Poor handwriting remains a significant problem in medicine. *J. R. Soc. Med.* **2006**, *99*, 645–646.
15. Holzinger, A.; Geierhofer, R.; Searle, G. Biometrical signatures in practice: A challenge for improving human-computer interaction in clinical workflows. In *Mensch & Computer: Mensch und Computer im Strukturwandel*; Heinecke, A.M., Paul, H., Eds.; Oldenbourg: Munich, Germany, 2006; pp. 339–347.

16. Lee, S.W. *Advances in Handwriting Recognition*; Series in machine perception and artificial intelligence; World Scientific: Taejon, Korea, 1999; Volume 34.
17. Holzinger, A.; Schloegl, M.; Peischl, B.; Debevc, M. Preferences of handwriting recognition on mobile information systems in medicine: Improving handwriting algorithm on the basis of real-life usability research. In Proceedings of the 2010 International Conference on e-Business, Athens, Greece, 26–28 July 2010; pp. 1–8.
18. Holzman, T.G. Computer-human interface solutions for emergency medical care. *Interactions* **1999**, *6*, 13–24.
19. Anantharaman, V.; Han, L.S. Hospital and emergency ambulance link: Using IT to enhance emergency pre-hospital care. *Int. J. Med. Informat.* **2001**, *61*, 147–161.
20. Baumgart, D.C. Personal digital assistants in health care: Experienced clinicians in the palm of your hand? *Lancet* **2005**, *366*, 1210–1222.
21. Chittaro, L.; Zuliani, F.; Carchietti, E. Mobile devices in emergency medical services: User evaluation of a PDA-based interface for ambulance run reporting. In Proceedings of the 1st International Conference on Mobile Information Technology for Emergency Response, Sankt Augustin, Germany, 22–23 February 2007; Springer-Verlag: Berlin/Heidelberg, Germany, 2007; pp. 19–28.
22. Holzinger, A.; Errath, M. Mobile computer Web-application design in medicine: Some research based guidelines. *Univers. Access Inform. Soc.* **2007**, *6*, 31–41.
23. Holzinger, A.; Basic, L.; Peischl, B.; Debevc, M. Handwriting recognition on mobile devices—State of the art technology, usability and business analysis. In Proceedings of the Proceedings of the International Conference on e-Business, Seville, Spain, 18–21 July 2011; pp. 219–227.
24. Klann, M.; Malizia, A.; Chittaro, L.; Aedo Cuevas, I.; Levialdi, S. Hci for emergencies. In Proceedings of the CHI '08 Extended Abstracts on Human Factors in Computing Systems, Florence, Italy, 7–10 April 2008; ACM: New York, NY, USA, 2008; pp. 3945–3948.
25. Lewis, J.R. Input rates and user preference for three small-screen input methods: Standard keyboard, predictive keyboard, and handwriting. In Proceedings of the Human Factors and Ergonomics Society 43rd Annual Meeting, Houston, TA, USA, 27 September–1 October 1999; pp. 425–428.
26. Haller, G.; Haller, D.M.; Courvoisier, D.; Lovis, C. Handheld *vs.* laptop computers for electronic data collection in clinical research: A crossover randomized trial. *JAMIA* **2009**, *16*, 651–659.
27. Balter, M. ARCHAEOLOGY: Paintings in Italian cave may be oldest yet. *Science* **2000**, *290*, 419–421.
28. Hess, L. *The History and Development of Handwriting from Prehistoric Times to 1925*; Butler University: Indianapolis, IN, USA, 1937.
29. Schenk, J.; Rigoll, G. *Mensch-Maschine-Kommunikation: Grundlagen Von Sprach- Und Bildbasierten Benutzerschnittstellen* (in German); Springer: Berlin, Germany, 2010; pp. 151–164.
30. Sesa-Nogueras, E.; Faúndez-Zanuy, M.; Mekyska, J. An information analysis of in-air and on-surface trajectories in online handwriting. *Cognit. Comput.* **2012**, *4*, 195–205.

31. Tappert, C.; Suen, C.; Wakahara, T. The state of the art in online handwriting recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **1990**, *12*, 787–808.
32. Kosmala, A. HMM-basierte Online Handschrifterkennung: Ein integrierter Ansatz zur Text- und Formelerkennung (in German); Ph.D. thesis, University of Duisburg-Essen, Essen, Germany, 2000.
33. Perwej, Y.; Chaturvedi, A. Machine recognition of hand written characters using neural networks. *CoRR* **2012**, abs/1205.3964.
34. Eden, M. Handwriting and pattern recognition. *IRE Trans. Inf. Theory* **1962**, *8*, 160–166.
35. Gao, Y.; Jin, L.; He, C.; Zhou, G. Handwriting character recognition as a service: A new handwriting recognition system based on cloud computing. *ICDAR* **2011**, doi:10.1109/ICDAR.2011.181.
36. Huang, B.; Zhang, Y.; Kechadi, M. Preprocessing techniques for online handwriting recognition. In *Intelligent Text Categorization and Clustering*; Studies in Computational Intelligence; Springer: Berlin/Heidelberg, Germany, 2009; Volume 164, pp. 25–45.
37. Tang, Y.; Qu, Y.; Suen, C. Entropy-reduced tansformation approach to pattern recognition. In Procceedings of the IAPR Workshop on CV, Tokyo, Japan, 12–14 October 1988; pp. 347–350.
38. Tang, Y.; Suen, C. Nonlinear shape restoration by transformation models. In Procceedings of 10th International Conference on Pattern Recognition, Atlantic City, NJ, USA, 16–21 June 1990; Volume ii, pp. 14–19.
39. Brakensiek, A.; Kosmala, A.; Rigoll, G. Comparing normalization and adaptation techniques for online handwriting recognition. In Procceedings of 16th International Conference on Pattern Recognition, Quebec City, Canada, 11–15 August 2002; Volume 3, pp. 73–76.
40. Guerfali, W.; Plamondon, R. Normalizing and restoring on-line handwriting. *Pattern Recogn.* **1993**, *26*, 419–431.
41. Taira, E.; Uchida, S.; Sakoe, H. Nonuniform slant correction for handwritten word recognition. *IEICE Trans.* **2004**, pp. 1247–1253.
42. Rehman, A.; Mohammad, D.; Sulong, G.; Saba, T. Simple and effective techniques for core-region detection and slant correction in offline script recognition. In Proceedings of the 2009 IEEE International Conference on Signal and Image Processing Applications (ICSIPA), Kuala Lumpur, Malaysia, 18–19 November 2009; pp. 15–20.
43. Kim, D.H.; Kim, E.J.; Bang, S.Y. A variation measure for handwritten character image data using entropy difference. *Pattern Recogn.* **1997**, *30*, 19–29.
44. Park, J.S.; Kang, H.J.; Lee, S.W. Automatic quality measurement of gray-scale handwriting based on extended average entropy. In Proceedings of the 15th International Conference on Pattern Recognition, Barcelona, Spain, 3–7 September 2000; Volume 4, pp. 426–429.
45. Cote, M.; Lecolinet, E.; Cheriet, M.; Suen, C. Automatic reading of cursive scripts using a reading model and perceptual concepts—The PERCEPTO system. *Int. J. Doc. Anal. Recogn.* **1998**, *1*, 3–17.
46. Kavallieratou, E.; Fakotakis, N.; Kokkinakis, G. New algorithms for skewing correction and slant removal on word-level [OCR]. In Proceedings of the 6th IEEE International Conference on Electronics, Circuits and Systems, Pafos, Cyprus, 5–8 September 1999; Volume 2, pp. 1159–1162.

47. Schenk, J.; Lenz, J.; Rigoll, G. Novel script line identification method for script normalization and feature extraction in on-line handwritten whiteboard note recognition. *Pattern Recogn.* **2009**, *42*, 3383–3393.
48. Goldsmith, D.E.; Collins, L.D.; Furches, E.C.; Kocienda, K. Handwriting Recognition Interface on a Device. U.S. Patent 0,226,091, 10 September 2009.
49. Assadollahi, R.O. Text Input System and Method Involving Finger-Based Handwriting Recognition and Word Prediction. EP 2 088 536 A1, 12 August 2009.
50. Bozinovic, R.; Srihari, S. Off-line cursive script word recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* **1989**, *11*, 68–83.
51. Guyon, I.; Schomaker, L.; Plamondon, R.; Liberman, M.; Janet, S. UNIPEN project of on-line data exchange and benchmarks. In Proceedings of the 12th International Conference on Pattern Recognition, Jerusalem, Israel, 9–13 October 1994; pp. 29–33.
52. Ballagas, R.; Rohs, M.; Sheridan, J.G.; Borchers, J. BYOD: Bring Your Own Device. In Proceedings of the Workshop on Ubiquitous Display Environments, Ubicomp, Nottingham, England, 7 September 2004.
53. Mauerer, J. Wie Unternehmen iPad & Co. nutzen. Available online: http://www.computerwoche.de/management/it-strategie/2367428/ (accessed on 7 October 2012).
54. Directive 2004/39/EC of the European Parliament and of the Council of 21 April 2004 on markets in financial instruments amending Council Directives 85/611/EEC and 93/6/EEC and Directive 2000/12/EC of the European Parliament and of the Council and repealing Council Directive 93/22/EEC.
55. Wagger, H. Die besten Innovationen auf der CeBIT (in German). Available online: http://social-apps-for-enterprises.tumblr.com/post/19231924050/die-besten-innovationenauf-der-cebit (accessed on 7 October 2012).
56. SoftNet Austria. Available online: http://www.soft-net.at (accessed on 14 November 2012).

© 2012 by the authors; licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/3.0/).