METRICS AND NEATENING OF HANDWRITTEN CHARACTERS

(Spine title: METRICS AND NEATENING OF HANDWRITTEN CHARACTERS)

Monograph

by

María Teresa Infante Velázquez

Graduate Program in Computer Science

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

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

THE UNIVERSITY OF WESTERN ONTARIO

SCHOOL OF GRADUATE AND POSTDOCTORAL STUDIES

CERTIFICATE OF EXAMINATION

Supervisor

_____
Dr. Stephen M. Watt

Examiners

_____
Dr. Robert Corless

_____
Dr. Eric Schost

_____
Dr. Roberto Solis-Oba

The thesis by

María Teresa Infante Velázquez

entitled:

METRICS AND NEATENING OF HANDWRITTEN CHARACTERS

is accepted in partial fulfilment of the
requirements for the degree of
Master of Computer Science

Date_____

_____
Chair of the Thesis Examination Board

ABSTRACT

At the present time, with the technology that many portable devices have, pen-based input is widely supported and input can be done through handwriting rather than just by typing. While character recognition and other input issues are the main consideration with these devices, there are many interesting output issues related to digital ink. This thesis investigates the questions of handwriting neatening and personal handwriting fonts.

We use the notion of "character metrics", which refer to the location and size of various features, such as ascender height and character width. Recording these features for handwritten characters allows for input and output transformations and may help with recognition in a 2-dimensional setting such as for mathematical handwriting recognition.

To be able to create personalized handwriting fonts, we have constructed a tool to collect samples where lines related to the characters metrics can be recorded. Handwriting always comes with variations in alignment and size so certain measurements based on these metrics can be extracted to perform the desired transformations to obtain a set of samples in a normalized form.

Keywords: font metrics handwritten characters neatening handwriting

ACKNOWLEDGMENTS

I owe my deepest gratitude to my supervisor Stephen Watt for giving me the opportunity to be part of his group of students, for giving guidance and advice through it all and for showing me that when you do what you are passionate about you are able to inspire others to do the same no matter what your field of study is.

I thank all the members of the Ontario Research Centre for Computer Algebra for offering me their advice and assistance which was very helpful and appreciated and for their encouraging words when needed.

I express my gratitude to all the staff in the computer science department for always being kind and friendly; for offering me advice and always answering to all my questions or pointing me to where I could find the answers.

Last but definitely not least I want to thank my family and friends. First and foremost to my mom, for always believing in me and giving me reassurance when I needed it. Without her unconditional support I would not have been able to get here. To my other mom, aunt and uncle for their encouraging words since I decided to pursue this. Also to my “brothers” who each helped me either reading the thesis or giving advice on how to write it. To my grandma for always looking out for me and not worrying about what I was doing (I miss you).

To my friends for their encouraging words, but specially to my friend for “being” with me every day since I moved and helping me whenever I got stuck, giving me a different point of view, hearing all about what I wanted to do, in other words for the enormous support.

Table of Contents

CERTIFICATE OF EXAMINATION ii
ABSTRACT iii
ACKNOWLEDGMENTS iv
CHAPTER 1 INTRODUCTION 1
1.1 Pen-Based Input 2
1.1.1 Liveboard 3
1.1.2 Inkboard 3
1.1.3 Electronic Chalkboard 4
1.1.4 MobileWrite 4
1.1.5 MyScript 5
1.1.6 RiteScript Software 6
1.1.7 Tablet PC 7
1.1.8 Windows Journal and OneNote 9
1.2 Handwriting Fonts 10
1.2.1 vLetter 10
1.2.2 Quantum Enterprises 12
1.2.3 Scanahand 13
1.2.4 Walter Ware 14
1.2.5 Font Generator 15
1.3 Other Work 15
1.3.1 Handwriting Interface 15
1.3.2 Local Slant Stimulation for Handwritten English Words 16
1.4 Thesis Organization 16
CHAPTER 2 FONT METRICS 18
2.1 Metric Lines and Measures 18
2.1.1 Baseline 19
2.1.2 Mean or X Line and Height 19
2.1.3 Ascender Line and Height 20
2.1.4 Top or Cap Line and Height 20
2.1.5 Descender Line and Height 21
2.1.6 Central Line 21

2.1.7 Width ... 23
2.1.8 Slant (angle) ... 23
CHAPTER 3 DIGITAL INK ... 24
3.1 JOT ... 24
3.2 UNIPEN ... 25
3.3 ISF ... 25
3.4 InkML ... 26
3.4.1 Elements ... 26
<ink> ... 26
<traceFormat> and <trace> ... 27
<traceGroup> ... 28
CHAPTER 4 ANNOTATING CHARACTER SAMPLES ... 29
4.1 Tool for Annotating Samples ... 31
4.2 Character Recognition for Automatic Analysis ... 34
4.3 Elimination of the slant ... 36
CHAPTER 5 HANDWRITING NEATENING ... 38
5.1 Symbol Variations ... 38
5.2 Partial Data from Metric Lines ... 42
5.3 Method ... 42
5.3.1 Determination of desired font size ... 42
5.3.2 Feature scaling ... 43
Scale ... 43
5.3.3 Conjugated feature scaling ... 47
De-slanting ... 47
Re-slant ... 49
5.4 Personalized Text ... 51
CONCLUSION ... 55
REFERENCES ... 57
CURRICULUM VITAE ... 60

Table of Figures

Figure 1. Screen shot of MobileWrite in Windows Mobile (left) and Palm OS (right) - images from [4] 4
Figure 2. Screen shoot of MyScript on Windows OS. 5
Figure 3. Screenshot of ritePen software - image from [6]. 6
Figure 4. Screenshot of riteForm software - image from [6]. 7
Figure 5. Screen shot of handwriting recognition in the Tablet PC Input Panel in Windows 7. . 8
Figure 6. Screen shot of features of Windows Journal 2007. 9
Figure 7. Form for vLetter Pro (cursive) - image from [11]. 11
Figure 8. Form for vLetter Pro (print) - image from [11]. 11
Figure 9. Example of vLetter handwriting font - images from demo in [11] 12
Figure 10. Example of changes made by the Scriptalizer Software - image from [12]. 13
Figure 11. Form to collect characters samples for Quantum Enterprises - image from [12]. 13
Figure 12. Form 1 to collect characters samples, provided by Scanahand version 3.0. 14
Figure 13. Form 2 to collect characters samples, provided by Scanahand version 3.0. 14
Figure 14. Base Line. 19
Figure 15. Mean line. 19
Figure 16. Mean Height. 19
Figure 17. Ascender line. 20
Figure 18. Ascender Height. 20
Figure 19. Top or cap line. 20
Figure 20. Cap or top Height. 20
Figure 21. Descender Line. 21
Figure 22. Descender Height. 21
Figure 23. Central Line. 21
Figure 24. Lines for Metrics. 22
Figure 25. Character Heights. 22
Figure 26. Width. 23
Figure 27. Inclination Angle or slant. 23
Figure 28. Example of InkML file – from [21]. 27
Figure 29. Example of <trace> and <traceFormat>. 28
Figure 30. Example of <traceGroup> - from [21] 28

Figure 31. Example of annotations on a lowercase character. 32
Figure 32. Example on annotations on an uppercase character. 33
Figure 33. Correct position for mean line. 34
Figure 34. Incorrect position for mean line. 34
Figure 35. Ideal approximation of line position, after character recognition (lowercase "p"). .. 35
Figure 36. Ideal approximation of line position, after character recognition (uppercase “P”). . 36
Figure 37. Example 1 de-slanting word. 36
Figure 38. Example 2 de-slanting word. 37
Figure 39. Characters supported in the application. 39
Figure 40. Example 1 of characters in Time New Roman. 39
Figure 41. Example 1 of characters in Agency FB. 40
Figure 42. Example 2 of characters in Times New Roman. 40
Figure 43. Example 2 of characters in Agency FB. 40
Figure 44. Ascender height for the application. 41
Figure 45. Character classification. 41
Figure 46. Variables from Formula B. 44
Figure 47. Variables from Formula C. 44
Figure 48. Variables from Formula D. 45
Figure 49. Variables from Formula E. 45
Figure 50. Variables from Formula F. 46
Figure 51. Original characters – example 1. 46
Figure 52. Original characters - example 2. 46
Figure 53. Neatened characters - slant ignored - example 1. 47
Figure 54. Neatened characters - slant ignored - example 2. 47
Figure 55. Variables from Formula H. 48
Figure 56. Capturing inclination of character. 49
Figure 57. Neatened characters - considering slant - example 1. 50
Figure 58. Neatened characters - considering slant - example 2. 50
Figure 59. Character zoom in. 51
Figure 60. Example of input word. 52
Figure 61. Example 1 of handwriting before and after neatening. 52
Figure 62. Example 2 of handwriting before and after neatening. 52
Figure 63. Example 3 of handwriting before and after neatening (uppercase). 53

Figure 64. Example 4 of handwriting before and after neatening (lowercase). 54
Figure 65. Example 5 of handwriting before and after neatening (numeric). 54
Figure 66. Example 6 of handwriting before and after neatening (lowercase). 54

CHAPTER 1

INTRODUCTION

Pen-based input is supported by many devices. With this, input is no longer just made through the traditional keyboard, keypad, mouse or touchpad. With the growing number of smartphones, PDA's, Tablet PC's and other touch sensitive devices, the capture of writing and drawing can be made using a stylus or even the tip of the fingers. The main capability of pen-based devices is interpreting entered computer actions via handwritten inputs and character recognition.

This feature makes the capture of handwritten input more natural for the user, since the process is very similar to writing on a piece of paper. It also allows the user to create diagrams or drawings more naturally than by using a mouse.

Taking into consideration that character recognition is a significant part of the majority of these existing handheld devices, it is worth mentioning that when character recognition is performed the recognized characters are typically displayed in a typeset font. None of the available fonts for the output text are based on personalized handwriting styles, so rendering the recognized text is a jarring and distracting event because of the large visual changes.

The principal objective of this thesis is to explore the creation of an application that can neaten handwritten characters as well as have the capability to create personalized handwritten fonts.

Character metrics are the basic concept in describing character dimensions. Metrics refer to the measures of the characters, the parameters used to locate where the characters are to be positioned and how they should be spaced and aligned.

Chapters two and three of this thesis give background. The information presented is relevant in order to have a better understanding of the topic of metrics, related topics of handwriting neatening and handwriting in general, and digital ink formats.

The rest of this thesis will cover in detail the method used for neatening the characters, as well as some problems identified in the output characters. A description of how the characters should be annotated, and how the process of neatening, could be implemented in character recognition to improve recognition rates.

1.1 Pen-Based Input

There is a wide variety of systems and software that use pen input. Each accepts handwriting as an input data form and has some capabilities such as sharing the written information between networks, saving handwritten input and character recognition. Below are brief descriptions of a few of the first systems and software that are hand writing enabled.

1.1.1 Liveboard

The Liveboard system [1], includes an application in which users can write and later retrieve documents. The input can be made through a cordless pen that allows remote pointing and input, and it also provides features such as variable brush sizes and an eraser. It also provides user interfaces for meetings, presentations and remote collaboration that can be used between different locations.

This Liveboard system [1] was developed in 1990 by the Xerox-Palo Alto Research Center.

1.1.2 Inkboard

Inkboard [2] is a Tablet PC software tool that accepts digital ink as an input and offers a set of communications options such as sharing ink strokes over a network as well as audio/video conferencing.

Inkboard is presented as a “visual thinking tool”, and as one of the most powerful tools known until 2004 in freehand sketching. It was used in designed disciplines such as architectural and graphic design.

The Inkboard application allows the creation of users using character recognition. This enables the storage and retrieval of digital sketches. Inkboard uses Ink Serialized Format (ISF) for stroke storage.

This network-sharing application was developed in the Intelligent Engineering Systems Lab and published by Hai Ning et al.

1.1.3 Electronic Chalkboard

Electronic Chalkboard [3] is also referred as E-chalk, this software system gives the user the opportunity to write using a stylus. The software emulates a classical chalkboard and provides other features such as adding images, activation of a computer algebra system and interactive Java Applets.

This board is enabled with "intelligent assistants". One provides the interpretation for handwriting input. Another assistant identifies mathematical formulas. Sketches of graphs are accepted as input data.

In handwriting recognition the E-Chalk determines the role of each symbol by using a spanning tree. The baseline is considered the most important feature of a formula. All the interpretations for the symbols are based on their relative position with the baseline.

This Intelligent Electronic Chalkboard [3] was developed at the Freie Universität Berlin (Germany), and the article was published by Gerald Friedland et al in 2004.

1.1.4 MobileWrite

Figure 1. Screen shot of MobileWrite in Windows Mobile (left) and Palm OS (right) - images from [4]

MobileWrite was created by Inkmark Software [4]. This company is involved in developing handwriting recognition software for Pocket PC's running Windows Mobile

and devices running Palm OS. The principal goal of MobileWrite is to enable text and data entry through pen input in devices such as phones and PDA's with touchscreen (that is for devices that may lack a keyboard or have an impractical one).

MobileWrite also provides the handwriting capability for programs such as calendars and address books, avoids the training process of other handwriting recognitions programs.

1.1.5 MyScript

Figure 2. Screen shoot of MyScript on Windows OS.

MyScript [5] takes into consideration each person's writing style, and it can interpret the different writing styles in different Western and Asian languages.

In the process of handwriting recognition, this program relies on three so-called "experts" in charge of the recognition process. These experts are: the character expert, which extracts features from individual characters, the segmentation expert, which analyzes where the strokes are going to be divided into characters or words and the last expert is the language expert which makes use of linguistic information. These three are combined and with their resources of linguistic material they provide a recognition result.

1.1.6 RiteScript Software

Ritescript [6] is part of Evernote Corporation, developers of software that involves digital ink and recognition technologies for desktop and mobile devices. RiteScript is also the name for their patented natural handwriting recognition technology, implemented now in various languages such as English, German or French. This technology is used in their ritePen and riteForm software and in a numerous digital ink features of the EverNote [7] software. With other recognition software packages the output is given in a typeset font, using the handwritten strokes just for input and with the possibility of saving the characters in their handwritten form.

Figure 3. Screenshot of ritePen software - image from [6].

The ritePen is a handwriting recognition software package for Windows OS, pen-enabled PC's and mobile devices. It claims to accept any handwriting style, to not require learning or training process and to allow writing on any part of the screen. The output text can be used in any Windows application.

Figure 4. Screenshot of riteForm software - image from [6].

The riteForm SDK is a multi-platform and multi-language handwriting recognition software package for pen-enabled devices and server integration. It is a form processing software package directed mainly to mobile enterprises and small businesses, providing specialized vocabularies and data templates.

Evernote is a multi-platform, multi-device package/service for creation of text and ink notes with images, videos, clips from webpages, and snapshots of whiteboards. Evernote [7] allows users to capture from real and digital lives providing the aid for easy find.

1.1.7 Tablet PC

A Tablet PC [8] provides a screen that may be written on directly by the user with a digital pen, either to control the PC or to input information as handwriting or drawings.

This process, also known as “inking”, enables users to add “digital ink” to a range of Windows applications. The digital ink appears as natural-looking handwriting on the screen. The digitalized handwriting can be converted to standard typeset text through handwriting recognition, or it can remain as handwritten text. Both the converted text in typeface and the cursive handwritten text function equally well as data formats in Windows applications and platforms.

On Tablet PCs, a digitizer overlaid on the LCD screen creates an electromagnetic field. The pen comes in contact with the screen's electromagnetic field. Its motion is detected on the screen as a series of data points. The digitizer collects information from the pen movement in a process called sampling, and is capable of sampling 130 "pen events" (units of motion that correspond to data points) per second. These pen events are then represented visually on the screen as pen strokes.

The inking process gives users the choice of converting the handwritten data to standard text through handwriting recognition. It can also preserve the data in its ink format.

Tablet PC provides a pen and a microphone, which were new choices for mass-market users, without eliminating the traditional devices that is the mouse and keyboard. Even though pen could be used as a substitute for the mouse, its widest use is for handwriting input.

The following image shows how the Tablet PC Input Panel available in Windows 7 Professional, goes from the handwritten word directly to typeset letters to present the recognition results.

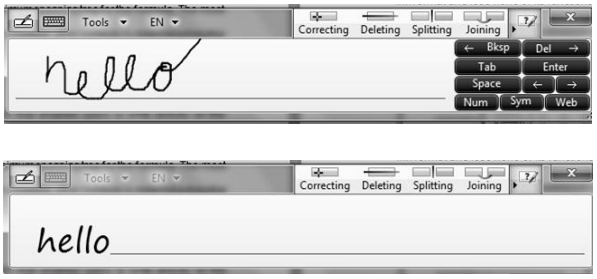

Figure 5. Screen shot of handwriting recognition in the Tablet PC Input Panel in Windows 7.

1.1.8 Windows Journal and OneNote

In the words of Paul Yao [9], "Windows Journal is an ink-enabled cross between Notepad and Paint". This description is highly accurate since the user can not only write (handwrite) but also can add images, erase, adjust stroke thickness, color, highlight and insert spaces between already written words. The editing feature of a document after being written is the main difference from when writing in actual paper.

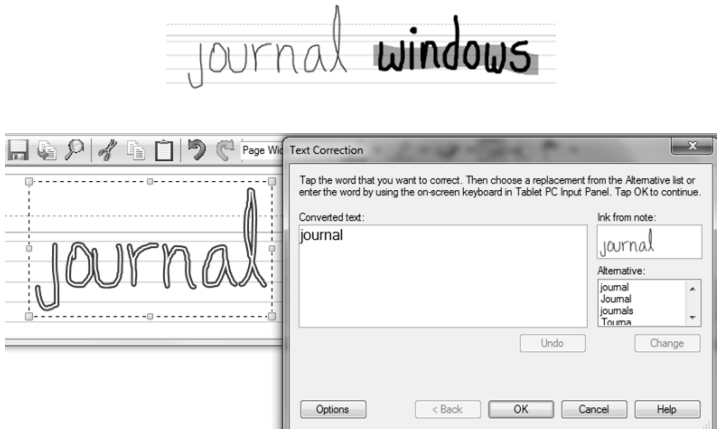

Figure 6. Screen shot of features of Windows Journal 2007.

In addition to these features, a note can be saved as images and, when requested, a handwritten word can be converted to typeset (handwriting recognition).

Even though Windows Journal has features to change the appearance of the strokes, similar to the Tablet PC Input Panel it does not involve doing any kind of handwriting neatening for the words. OneNote [10] is a package more complex than Windows Journal, and has more input information like organized text, handwriting (digital), video, audio, picture among others. Nevertheless, neither does a change to the handwritten characters.

1.2 Handwriting Fonts

We pass from creating typeset fonts to give documents a formal look, to creating handwriting fonts to make a note, an e-mail or even make a letter feel more personal.

Pen-based devices, despite being able to capture information through a stylus, do not have as their main purpose preserving the information in handwritten characters. Character recognition is highly used and it provides the identification of handwritten characters, giving the resulting output characters displayed in a typeset font.

Taking the above into consideration, there are numerous fonts that were created based on handwriting styles and can be found in various websites such as free-fonts.com, fontspace.com and 1001freefonts.com. Regardless of being based on handwritten characters these do not preserve the specific characteristics of each user's particular handwriting style.

There are some companies dedicated to the creation of personal handwriting fonts. We highlight some below with the characteristics that each provide.

1.2.1 vLetter

In 1988 Signature Software, Inc. was founded. From 1988 to 1992, the company developed and patented some technology to produce custom handwriting fonts and cursive handwriting fonts that are a reproduction of handwriting styles.

In 1992 the Personal Font was released, software to create fonts reproducing the handwriting style of the users. It was the only one that provided custom-built handwriting fonts with context-sensitive modifications. An example for context modification is when

a version of a character is used when it is written at the end of a word, and other, when it is in the middle or at the beginning of the word. In 1993 PenFont was released. It provided custom-built fonts, similar to Personal Font, with the addition of keyboard characters in the handwritten font. The International version of PenFont provided European accented characters.

In 2000, the company changed its name to vLetter, Inc., to signify "virtual letter". Their product names were updated from Personal Font to vLetter Pro and from Penfont to vLetter Print.

vLetter Pro creates handwriting fonts with the help of forms to collect the handwriting samples. The samples consist of all the letters in different words and combinations of letters (see Figure 7 for partial image), to be able to reproduce the natural variations that occur in cursive handwriting. For print handwriting the samples are letters and symbols (see Figure 8 for partial image). In both cases the only guideline used is the baseline.

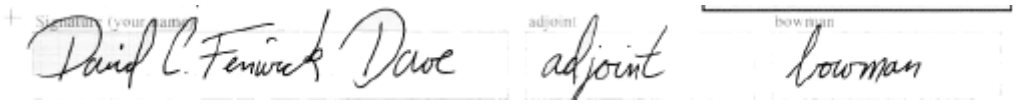

Figure 7. Form for vLetter Pro (cursive) - image from [11].

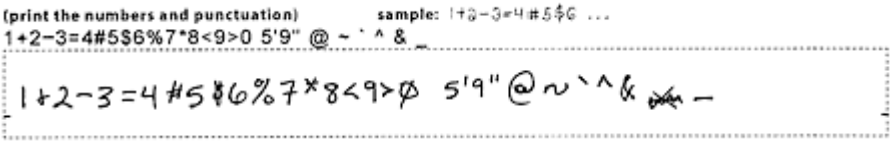

Figure 8. Form for vLetter Pro (print) - image from [11].

vLetter offers real connections between letters instead of overlapping the characters to obtain more accurate cursive handwriting fonts that represent better the natural handwriting. To accomplish this they make use of four different samples for the lowercase letters to capture the connections and variations particular to each handwriting style. Figure 9 shows the demo of a font provided by vLetter in which the width of the stroke can be manipulated as well as the size of the font, before the font is obtained.

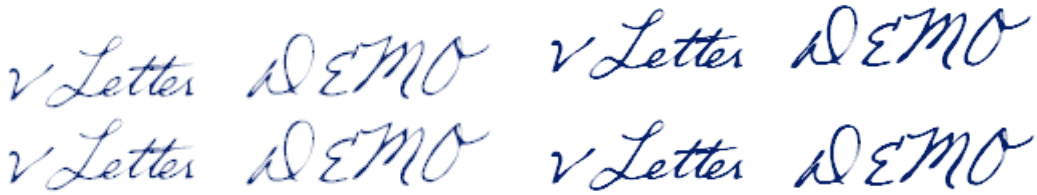

Figure 9. Example of vLetter handwriting font - images from demo in [11].

1.2.2 Quantum Enterprises

Quantum Enterprises [12] is a British company that offers the service of creating a personal handwriting font. On their product site, the company claims "The shape and joining strokes of a letter are dependent on the letter which came before it, and the letter which comes after… you have to accept these limitations and understand that your handwriting font will never look EXACTLY like your actual handwriting".

After the quote above, this company offers an "ultra" service that includes the capture of more characters and some words to get the connection between words. It also offers the Scriptalizer Software which uses "mistakes" to enhance the personal handwriting font. It replaces some characters with different versions as shown in Figure 10 to give the font a more natural effect.

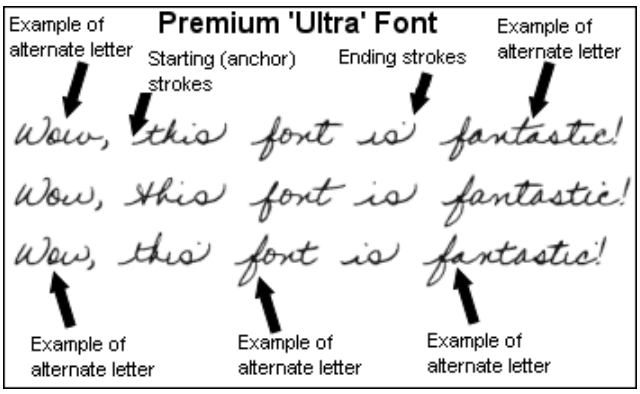

Figure 10. Example of changes made by the Scriptalizer Software - image from [12].

To capture samples of the characters for the creation of the handwriting font, they rely on characters written in a form (See Figure 11) that uses only the baseline as a reference.

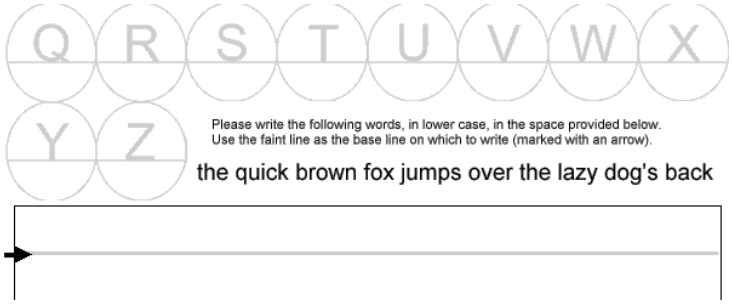

Figure 11. Form to collect characters samples for Quantum Enterprises - image from [12].

1.2.3 Scanahand

The Scanahand software is developed by High-Logic B.V [13]. It uses two forms to create the handwriting font. The first form is for letters of the alphabet, numbers and most common symbols (see Figure 12). The second form contains accented characters generally used in other languages such as French or German (see Figure 13). Marks for various metrics, such as the ascender line, x or mean line, base line and descender line are provided as a guide (these terms are explained in Chapter 2). Marks can be extended into lines in the form to provide better guidelines to the user (see Figure 13). There is no

alteration than can be done to the character traces, other than the thickness of the stokes to render the font in bold.

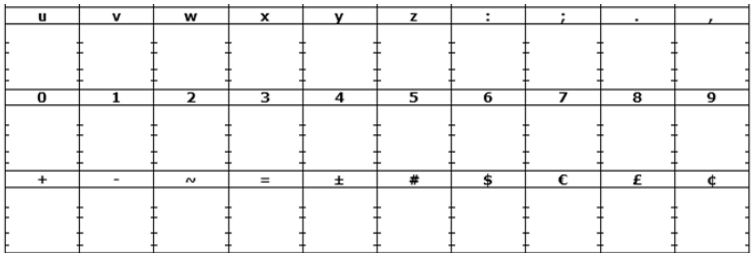

Figure 12. Form 1 to collect characters samples, provided by Scanahand version 3.0.

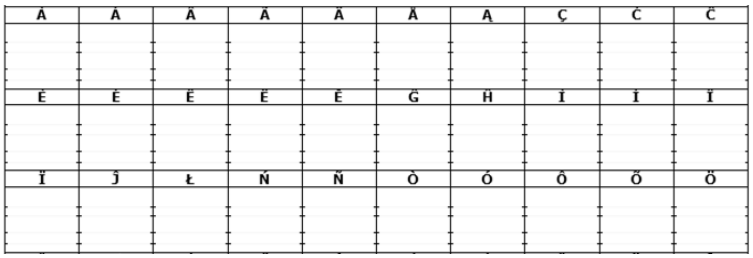

Figure 13. Form 2 to collect characters samples, provided by Scanahand version 3.0.

1.2.4 Walter Ware

Walter Ware [14] offers a service to create custom handwritten fonts. These handwritten fonts can be created based on cursive or print handwriting. The requirements for capturing the character samples are similar to the ones used by vLetter, which has a similar service. Considering the above, the required samples consist of individual characters and some words on white paper. There are no metric lines involved, not even the baseline. A randomizer program is provided along with the handwritten font. This changes the appearance of the ligatures between characters to accomplish a natural, varied look in the printed output.

1.2.5 Font Generator

The Font Generator program [15], as the name suggests, creates fonts. These are based on handwritten samples. It is an on-line service where the fonts are generated in minutes. Therefore the resulting characters are not as precise as the ones provided by the other handwriting creators mentioned above, nor of the same quality. The way the character samples are captured is through forms. These make use of the top line, mean or x line, base line and descender line. By the above, the similarity in the characters' height depends entirely on the precision that the user can accomplish.

1.3 Other Work

1.3.1 Handwriting Interface

The concept of handwriting neatening is used by H. Ocamura et al in the paper "Handwriting Interface for Computer Algebra Systems" [16] and is related to the development of a recognition system for handwritten mathematical expressions. The general process for the system described begins by rewriting the characters in a neat manner, and then these neatened characters are transformed in the corresponding mathematical expression in typeset.

Their approach uses character recognition on each character that forms the formula, then performs the handwriting neatening before finishing the process by translating the formula to its corresponding typed form.

1.3.2 Local Slant Stimulation for Handwritten English Words

Y. Ding et al [17] talk about the slant in handwriting and how it is usually calculated as an average value for a word. They propose to use local estimation of the slant by calculating the slant for each of the characters instead of the general average for the whole word. This was done for character recognition purposes. We use a similar approach by capturing the value of the slant from each character while they are being individually captured.

1.4 Thesis Organization

The main goal of our work is to accomplish handwriting neatening for hand printed text. We assume the characters are well isolated. We have developed a cross-platform framework for this purpose. A generic description of how the platform works is as follows: It collects a sample of each character that forms the English alphabet and it is saved for later retrieval. A word is typed and after processing it the neat version of the word is displayed by the platform in the handwritten font selected.

Chapter 2 and 3 provide the basic background about font metrics and digital ink. Chapter 2 reviews the metrics that are related to typographic fonts and which we also use for hand writing samples. Chapter 3 reviews different formats to store digital ink used by existing devices.

Chapter 4 describes how our application can be considered as a tool for annotating samples. It also gives some suggestions on how to improve the application, some possible difficulties and how they might be overcome.

In Chapter 5 we explain how some characters provide partial information and the existing variations among symbols. A detailed explanation of the method used to neaten the collected characters. It also illustrates the differences between the characters before and after the process.

CHAPTER 2

FONT METRICS

This Chapter gives a description of font metrics and each of the metric lines used in the proposed application. These lines were taken from font metrics used in typography and adjusted to the handwritten characters used here. We also explain each of the different measurements that are calculated with the metric lines, and why these are taken in consideration for the proposed method.

2.1 Metric Lines and Measures

There is widespread agreement on the measurements used to characterize typeset symbols and these are familiar to the creators of graphical user interfaces. We have therefore applied these measurements to handwritten characters as well. These are the *ascender, top, mean, central, base* and *descender* lines. All of these will be describe in this section.

To perform any transformation on the captured characters, it is necessary to extract specific information from each of the letters. This data collection is also based in the font analysis used in typography.

The basic measurements required to perform the needed neatening and reallocation of the captured characters are the following: ascender height, top / cap height, mean / x height, descender height, width and slant (angle).

2.1.1 Baseline

Figure 14. Base Line.

The *baseline* is the imaginary line on which most of the characters are based. Generally, contiguous characters retain their alignment using the baseline as reference. Although certain characters extend below or above this line, it continues as the imaginary base.

This line can be considered as the most important, since it is the guide for writing and it is used also as reference to obtain all the heights of the characters.

2.1.2 Mean or X Line and Height

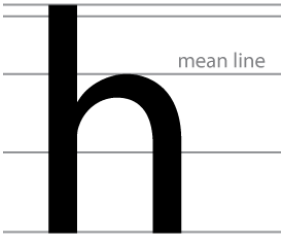

Figure 15. Mean line.

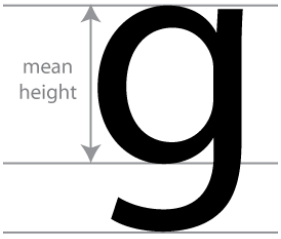

Figure 16. Mean Height.

The *mean line* is used to define the height (also called *mean height*) of the lowercase letters and it is located over the baseline. It is important to mention that not all the lowercase letters lie only between these two lines, some extend above the mean line or below the baseline. The characters that fall in those circumstances continue to take the mean line as a reference, since it provides the information to obtain their height.

Examples of characters that have mean height are “a”, “o” or “m” just to mention a few. The *mean height* is the distance between the *baseline* and the *mean line.*

2.1.3 Ascender Line and Height

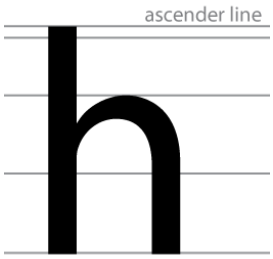

Figure 17. Ascender line.

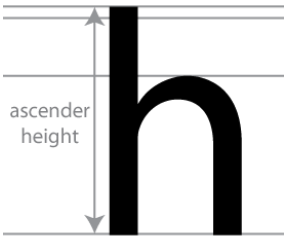

Figure 18. Ascender Height.

The *ascender line* helps to measure the height of lowercase letters (*ascender height*) which go beyond the mean / x-line, as in the case of the letters “l”, “h”, “t”, to name a few. Not to be confused with the line that it is used to calculate the height of capital letters, this will be addressed next. This line generally is located above all the lines used in character metrics. The *ascender height* is the space between the *baseline* and the *ascender line*.

2.1.4 Top or Cap Line and Height

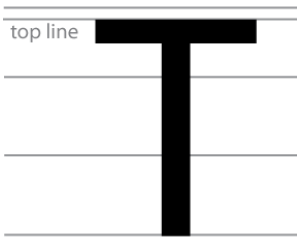

Figure 19. Top or cap line.

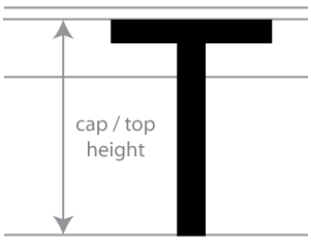

Figure 20. Cap or top Height.

The *top* or *cap line* is the line that is used to calculate the height of uppercase characters, which goes from the *baseline* to the *top line.* Usually in typography this line is located

below the ascender line, but it is not limited to that location. Having the word “top” as part of the name for this line can be confusing in relation to its position. Nevertheless, it can alternate positions with the *ascender line*. The space limited by the *top/cap line* and the *baseline* is usually known as *top height.*

2.1.5 Descender Line and Height

Figure 21. Descender Line.

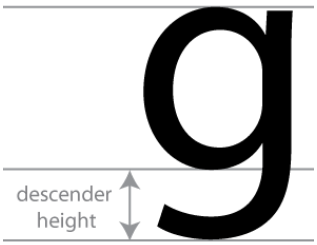

Figure 22. Descender Height.

The *descender line* is located underneath the base line. It is useful for characters such as the letter “j”, “g” or “y” among others, as it helps to mark the distance that these type of characters descend below the *baseline*.

The height refers to the distance between the *baseline* and the *descender line* and not to the whole character like in the previously mentioned heights.

2.1.6 Central Line

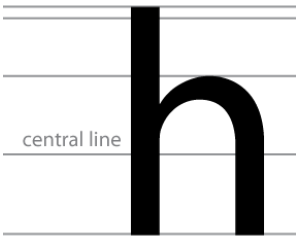

Figure 23. Central Line.

The *central line* is located between the *mean line* and *baseline.* This line becomes useful when the characters do not make contact with any other line. This line gives useful

information so we can determine the position of certain symbols. One example of this is the minus symbol, equality sign, a hyphen, since they do not make contact with any other line.

Figure 24 illustrates all the lines along with some representative characters to exemplify their use and Figure 25 contains all the heights provided by these lines.

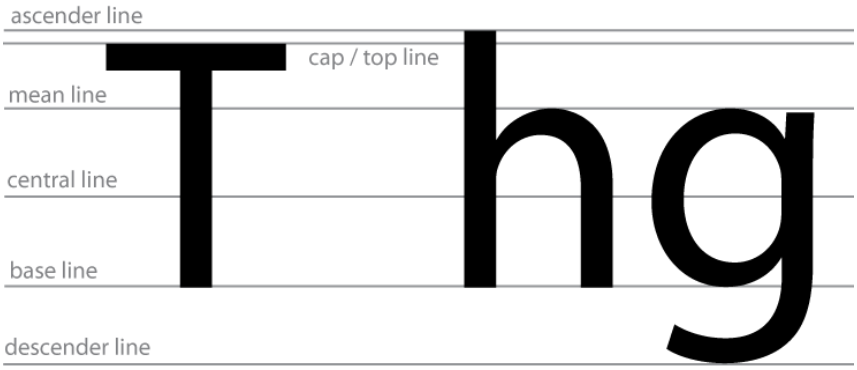

Figure 24. Lines for Metrics.

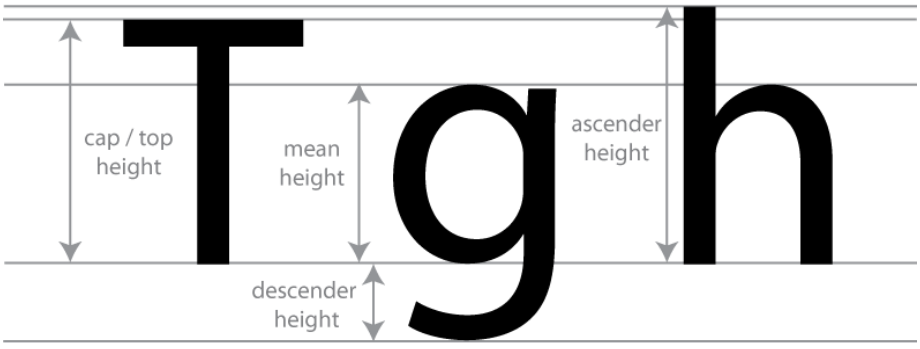

Figure 25. Character Heights.

2.1.7 Width

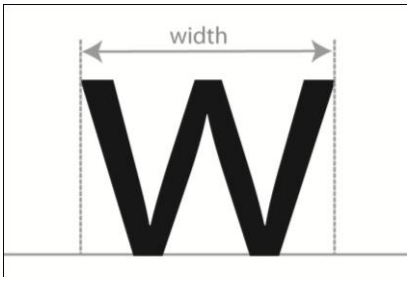

Figure 26. Width.

The *width* is the measure that goes from the first point of the character on the left to the last point on the right side of the character.

2.1.8 Slant (angle)

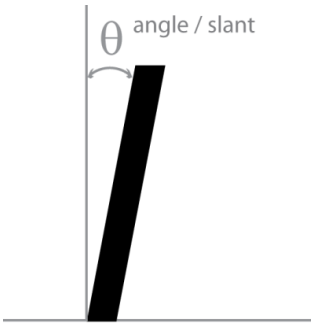

Figure 27. Inclination Angle or slant.

When the user writes with certain inclination either to the right or to the left, it is important to take that in consideration. The degree of that particular angle is referred to as the "slant".

CHAPTER 3

DIGITAL INK

Digital Ink technology has been evolving over the years and has become more popular. Part of this is due to the fact that it provides more natural ways of capturing input information. Some devices that use this type of technology are Tablet PC's and PDA's.

These portable devices store the information acquired through their particular digital ink applications in various types of formats.

3.1 JOT

In 1992, as an attempt to fulfill the main necessity of sharing ink-based information, several companies such as Slate, Lotus, GO, Microsoft, Apple and others compiled the result of their joint work in this format. It provided the availability to store electronic ink as well as sharing it between software packages.

Nevertheless, this format did not accomplish complete manipulation of digital ink, for example it is unable to modify the thickness of strokes or change colors.

Jot [18] is platform independent. This implies that ink created on a device can be shared with one that does not have the capability for pen. The format can be displayed on any graphics capable system.

3.2 UNIPEN

UNIPEN [19] is another digital-ink format. In this ink format, some features of the ones created by companies such as IBM, Apple, Microsoft, Slate, HP, AT&T, NICI, GO and ICI were successfully incorporated. After all this compilation work the UNIPEN saw daylight in 1994.

The main use for this format was in the scientific research field with the purpose of a common data format to facilitate data exchange. This was well used in data collection for handwriting recognition but did not have the flexibility required for other applications.

3.3 ISF

Ink Serialized Format is a graphic format used by ink-enabled programs when ink is stored. The acronym ISF [20] comes from Ink Serialized Format. This data type was created by Microsoft to store ink data in Windows platform devices. Unfortunately its use was limited to mobile devices, such as Tablet PC, PDA's or mobile PC's.

This format can be saved in XML or binary format. It can also be converted to a Graphic Interchange Format (GIF) with embedded ISF data which makes the ink accessible to non-ink-enabled programs.

ISF data has four types of encoding: ISF stream, GIF image with ISF stream embedded as metadata, ISF stream base-64 data in XML and GIF stored in a similar way as data in base-64.

3.4 InkML

InkML [21] stands for “Ink Markup Language”, created by the World Wide Web Consortium (W3C [22]). InkML is a format used for ink representation when the data is entered through and electronic pen or stylus. It is an XML-based language and makes a simple platform-neutral data format available. This in return facilitates the interchange between software applications of digital ink.

This is the first open format for digital ink suitable for both archival and streaming applications.

3.4.1 Elements

<ink>

Documents in InkML are well-formed XML documents that follow certain syntax rules. Each InkML document has all its content in a single <ink> element.

This element is the root for any InkML instance. The sub-elements of the <ink> element <trace>, <traceformat>, <tracegroup>, <context>, <annotation> and <annotationXML> they may be given in any order and can occur any number of times. When InkML is combined with other XML elements within applications a qualifier must be used in order to disambiguate the elements from different namespaces.

```
<ink>
   <trace>
     10 0, 9 14, 8 28, 7 42, 6 56, 6 70, 8 84, 8 98, 8 112, 9 126, 10 140,
     13 154, 14 168, 17 182, 18 188, 23 174, 30 160, 38 147, 49 135,
     58 124, 72 121, 77 135, 80 149, 82 163, 84 177, 87 191, 93 205
   </trace>
   <trace>
     130 155, 144 159, 158 160, 170 154, 179 143, 179 129, 166 125,
     152 128, 140 136, 131 149, 126 163, 124 177, 128 190, 137 200,
     150 208, 163 210, 178 208, 192 201, 205 192, 214 180
   </trace>
   <trace>
     227 50, 226 64, 225 78, 227 92, 228 106, 228 120, 229 134,
     230 148, 234 162, 235 176, 238 190, 241 204
   </trace>
   <trace>
     282 45, 281 59, 284 73, 285 87, 287 101, 288 115, 290 129,
     291 143, 294 157, 294 171, 294 185, 296 199, 300 213
   </trace>
   <trace>
     366 130, 359 143, 354 157, 349 171, 352 185, 359 197,
     371 204, 385 205, 398 202, 408 191, 413 177, 413 163,
     405 150, 392 143, 378 141, 365 150
   </trace>
</ink>
```

Figure 28. Example of InkML file – from [21].

<traceFormat> and <trace>

The element that describes the format for every trace recorded is the <traceFormat>. The complexity of the recorded data will depend of the capability of the selected device. The simplest form will specify the X and Y coordinates. These formats can go as far as providing tip force in the stylus or the state of the side buttons.

The <trace> is considered the fundamental data element in any InkML file and represents the actual trace data. This data is a sequence of ink points that are in a continuous form. The <trace> records the data captured by the digitizer. The encoding of the sequence of data is according to the specifications that <traceFormat> provides, it gives a meaning to the coordinate values. A sequence of traces will form units, such as characters, words or diagrams.

If a <traceFormat> is not specified, the encoding format by default would be X and Y coordinates. Traces might contain different types of information, depending on the

devices that generated them or the application that may use them. InkML delimits channels to identify the data encoded in a trace.

A channel can be *regular*, meaning that a value is recorded at every point of the trace, or *intermittent*, meaning that the value will not necessarily be recorded in every point of the sample. An example of regular are X and Y coordinates and the state of the side buttons of a stylus will be an intermittent channel.

```
<traceFormat id="XYP">
    <channel name="X" orientation="POSITIVE" type="DECIMAL"/>
    <channel name="Y" orientation="POSITIVE" type="DECIMAL"/>
    <channel name="P" orientation="POSITIVE" type="DECIMAL"/>
</traceFormat>

<trace>
     107.0 211.0, 107.0 209.0, 120.0 188.0, 128.0 178.0, 138.0 168.0,
     151.0 154.0, 163.0 141.0, 173.0 131.0, 182.0 122.0, 188.0 114.0,
     193.0 110.0
</trace>
```

Figure 29. Example of <trace> and <traceFormat>.

<traceGroup>

The element <traceGroup>, as the name indicates, is used to group together traces that have the same characteristics. The <traceGroup> element collects <trace> or other <traceGroup> elements into a single unit. Ink traces can also be grouped structured collections to reflect semantic structure.

```
<traceGroup brushRef="#penA">
   <trace>166.0 103.0, 166.0 105.0, 166.0 109.0, 166.0 113.0</trace>
   <trace>166.0 119.0, 167.0 121.0, 168.0 122.0, 170.0 122.0</trace>
</traceGroup>
```

Figure 30. Example of <traceGroup> - from [21].

CHAPTER 4

ANNOTATING CHARACTER SAMPLES

Our main goal is to devise a method to neaten handwriting, preserving the characteristics of each writer's individual style as well as the purposeful variations that occur within each handwriting sample. There is no formal specification for “neat” handwriting. By neatening we mean that the characters will be more uniform in size and layout, while retaining some natural variations. An ideal tool would be one that performs the neatening of the characters automatically right after they are written, with no previous capture of samples required.

Constructing such a system would be a great challenge, due to handwriting ambiguity and required computation. The latter will be discussed in more detail later in this chapter. In order to be able to neaten handwritten characters, the first thing we need to do is gather information about them. Our approach has been as follows: We take a handwriting sample with examples of each character and annotate each symbol with markers to locate the metric lines (e.g. base line, x-line, and slant). We use these measures to determine average values over the entire symbol set. This captures one aspect of the writers' style. We can then use these values to adjust the shapes of subsequently written symbols, by scaling parts of the symbols to match the desired metrics.

The various measures, with the exception of the width, depend on the case of the character and also, on the characteristics of each case. For example, when a character is a

lowercase letter, it must have a *mean height* and it can also have *descender* or *ascender height*. Letter “c” is an example of a character with only *mean height,* while “g” has *ascender height* in addition to *mean height*. And “b” has *mean height* and *ascender height.*

These measures are obtained from all the captured characters. The specification of which characters are available in the tool we developed are explained later on in Chapter 5. An average of each metric height and width is calculated. Then, to neaten a handwriting sample, the characters are scaled according to the standard average values of the metrics.

To make a handwritten character conform to the computed metrics, the different parts of the character are scaled linearly. If used naively, this method has the problem of modifying the slant non-uniformly over the character. For example, the neatened version of character “b” can present a slant in the *mean height* region and a different degree of inclination in the *ascender height.* A different scale is used with each height to get the average value of each.

When the ascender part is scaled to decrease the size, the degree of inclination is reduced and when the height is incremented the slant is also incremented. As a result, the character has two different slants, one in the ascender and another one in the mean height. Thus, it alters the original degree of slant of the original handwritten style. To overcome this problem we de-slant the characters before scaling them and re-slant them afterwards. With this, the angle of inclination remains the same as the original handwriting characters.

The overall design of our developed tool is as follows. The characters are captured individually to locate the different metric lines according to each character. We rely on the user to determine the correct position of each metric line. Measurements are stored

along with the strokes of the characters. Once all the handwritten characters and their average measures are stored, the user types words that are to be displayed in the user's handwriting style. The handwritten characters, corresponding to the typed words, are scaled according to the calculated average measures by the developed tool and displayed. Therefore, the displayed characters are considered to be neatened characters.

Making the process of neatening completely automatic without requiring any intervention of the user is challenging. One possible solution that may help overcome this is that the tool would determine the position of different metric lines, such as *baseline* and *mean line*, and calculate the height for later retrieval. This might be accomplished using character recognition to help with the determination of the required metric lines. More detail about this is given later on this Chapter.

4.1 Tool for Annotating Samples

We have created an application to allow human guided character sample analysis. The intervention of the user is required to enter the (handwritten) samples, and also to make adjustments to the corresponding metric lines. This is the best way to extract the necessary information.

The process to capture character samples is explained with the following steps:

1. The user enters the name for the handwriting style and creates a new style.

Steps 2 to 4 are repeated for each character in Figure 39.

2. The user traces the character on the canvas in the application.
3. Next, the metric lines are adjusted by the user according to the particular character. These metric lines will depend on the case of the letter. For example,

when the character is a capital letter, the *top* and the *base lines* are measured. However, in the case of lowercase letters the ascender, *mean* and *descender lines* are measured and not the *top line*.

After all the characters are captured, the output text can be displayed in one of the handwritten fonts that are stored by the user. So far the application can handle single words only.

The Figure 31 shows the required lines and positions, when the character to be stored is a lowercase letter like “a”. Such letters have a *mean height* only.

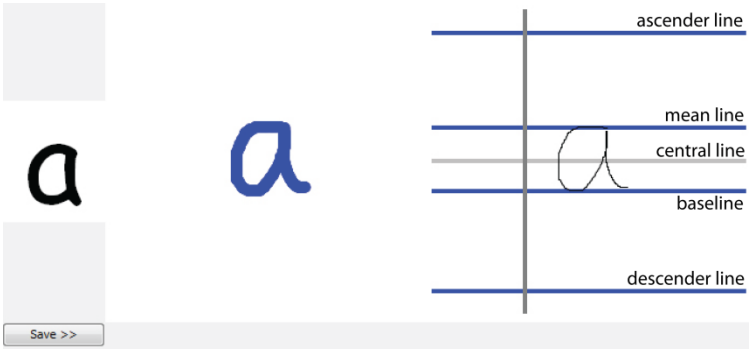

Figure 31. Example of annotations on a lowercase character.

Figure 32 illustrates the metric lines that need to be adjusted when the character is in capital letters. This case applies for all the twenty six characters that are part of the English alphabet.

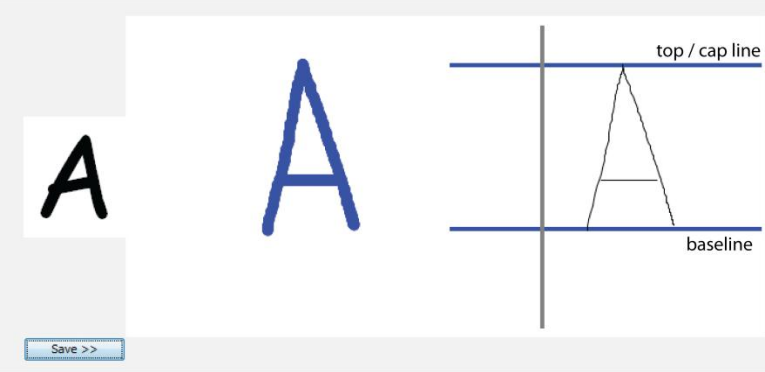

Figure 32. Example on annotations on an uppercase character.

Automatic detection of the metric lines is an improvement that may provide the user with a gentler experience. However, it also represents a great challenge considering that handwritten characters can have ambiguous positions for the lines, related to the metrics of each.

For instance, in the case of the character “h” the position for the mean line illustrates the ambiguity referred. To explain this ambiguity, we look at the letter ‘h’ and letter ‘a’ together.

However, it must be noted that the application allows the user to trace only one character at a time. The letter “a” is there just for reference.

Suppose the rule for automatic detection stated that the first point to be encountered when moving from the top downward was the position for the mean line. Then, the mean line calculated in case of ‘h’ would be wrong, since ‘h’ has also an ascender part. In the case of ‘a’, however, the correct mean line would be calculated since it stretches from base to mean line only. In Figure 33 an ideal calculation for the positions is presented and in Figure 34 an incorrect approximation for the position of the mean line is shown.

A similar problem regarding the automatic detection of positions for the metric lines is encountered with the characters that have a descender part. The ambiguity is specifically between the position for the *baseline* and the *descender line*. An example for this are the letters “c” and “g”. It is more challenging to detect the position of the *baseline* in the case of letter “g” since it also has a *descender line*. These two particular cases are just examples to showcase the challenges represented by the automatic detection. Therefore, a possible algorithm for automatic detection could include character recognition.

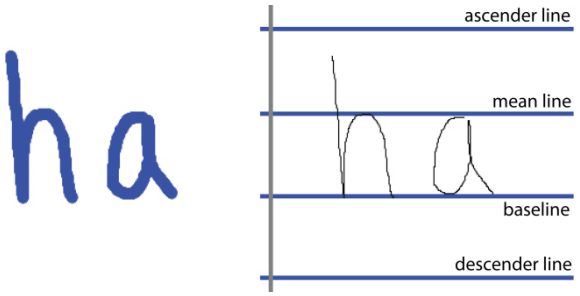

Figure 33. Correct position for mean line.

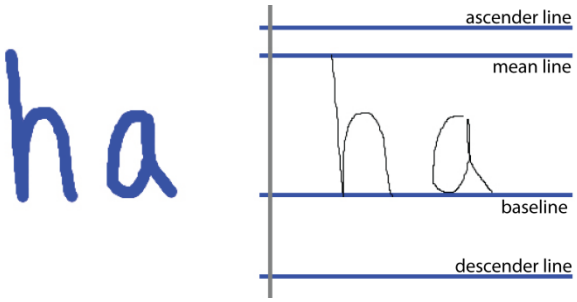

Figure 34. Incorrect position for mean line.

4.2 Character Recognition for Automatic Analysis

The work by H. Okamura et al [16], uses character recognition on handwritten mathematical formulas. The paper states that the recognized characters are treated with some transformations (neatening). However, details on the transformations are not given

in this paper. After the characters are transformed, character recognition is again performed on these transformed characters to change the handwritten formula to the corresponding one in the typeset form.

Based on the main idea of the paper above, a possible approach that could help accomplish the automatic accommodation of the metric lines, is to make use of character recognition on the captured handwritten character. After the character is recognized, the lines will be placed in a calculated position, according to the characteristics of each character. Ideally, no adjustments regarding the position of the metric lines will be required from the user.

This proposed theory of automatic analysis can be illustrated by considering the character 'p' as the letter to annotate. The input character will be recognized as uppercase or lowercase and the lines will be aligned accordingly. If the letter is recognized as the letter “p” (lowercase) the approximation of the locations of the lines would be according to the result of the recognition. This is shown in Figure 35. Minimal or possibly no adjustments will be needed by the user. A similar example would be with the letter “P” (uppercase). It is displayed in Figure 36. Both cases given by the automatic analysis represent an ideal result.

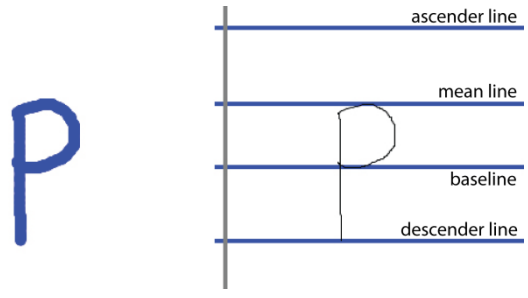

Figure 35. Ideal approximation of line position, after character recognition (lowercase "p").

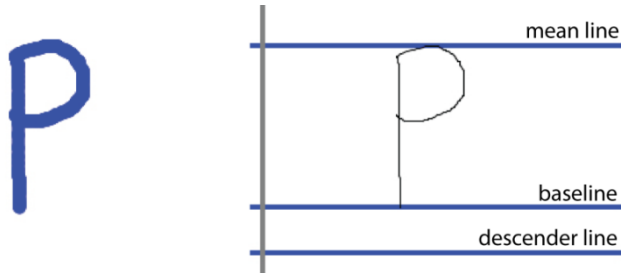

Figure 36. Ideal approximation of line position, after character recognition (uppercase "P").

4.3 Elimination of the slant

Another procedure that could be conducted in order to get more neatened words, is eliminating the slant. Since the goal of this tool is to have personal handwritten fonts, altering the natural slant of the characters could eliminate some of the personal characteristics that each style normally has, so we do not do it. However, judging by the results in Figure 37 and Figure 38 de-slanting the characters does not seem to cause a drastic change in the shape of the traces. This means the shape is not being altered excessively. As mentioned above it can always be considered as an option to remove the inclination of any handwriting style.

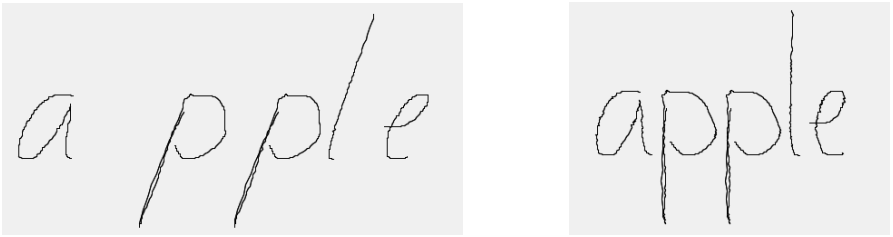

Figure 37. Example 1 de-slanting word.

Figure 38. Example 2 de-slanting word.

CHAPTER 5

HANDWRITING NEATENING

This Chapter describes the details of the main ideas that we have investigated for handwriting neatening and how the differences among characters prevent collecting the same measurements from all characters. It contains how the desired measures for the "new" neatened characters are calculated. The details of the method used are given along with the specifics of how the required measures were obtained.

5.1 Symbol Variations

Several symbols were taken in to account in addition to the twenty six letters that are part of the English alphabet. We considered a character set consisting of eighty five symbols. These symbols can have a wide variety of measures related to typography. The measurements taken for the personalized handwritten fonts, as mentioned before, are based in the metrics used on this field.

The characters and symbols considered in this application include the twenty six letters in the English alphabet, both in lowercase and uppercase, ten digits and twenty three symbols. All of them are listed in Figure 39.

a	b	c	d	e	f	g	h
i	j	k	l	m	n	o	p
q	r	s	t	u	v	w	x
y	z	A	B	C	D	E	F
G	H	I	J	K	L	M	N
O	P	Q	R	S	T	U	V
W	X	Y	Z	0	1	2	3
4	5	6	7	8	9	!	?
#	$	%	&	/	\	@	[
]	{	}	(	)	|	+	*
<	>	-	=	_			

Figure 39. Characters supported in the application.

The symbols such as parenthesis "(", ")" or at "@", just to name three, can have very different measures depending on the typesetting used for display. To illustrate this, we took the fonts Agency FB and Times New Roman as font examples. In the Times New Roman font, the height of these characters goes from the *descender line* to the *ascender line* (Figure 40). In this font the *ascender line* is located over the *cap/top line.* Characters like parenthesis "(" cover letters that are over the *mean line* like "l", "P" or "t", as well as the ones that go below the *baseline*, such as "g" or "j". The symbol at, "@", has the same height as the parenthesis, "(".

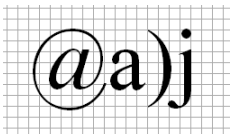

Figure 40. Example 1 of characters in Time New Roman.

In the Agency FB font the height of the "@" symbol goes from the *baseline* to the *cap/top line* (Figure 41). In this particular font the *ascender* and the *cap/top line* are at the same

level, with the exception of the character “t”. The parenthesis “)” stretches from between the *descender line* and the *baseline* to the *ascender line* (Figure 41) covering almost entirely all capital and lower case characters.

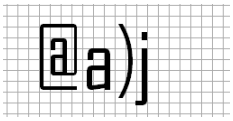

Figure 41. Example 1 of characters in Agency FB.

There are also other characters such as diagonal “/” or back slash” \” that instead of going from the *descender line* to the *ascender,* are limited to the height between the *baseline* and the *ascender*. These cases are the same for both fonts illustrated in Figure 42 and Figure 43.

Figure 42. Example 2 of characters in Times New Roman.

Figure 43. Example 2 of characters in Agency FB.

The heights that we considered in each set of symbols are described as follows:

The *ascender height* is the one that goes from *mean line* to *ascender line* instead of going from *baseline* to *ascender* like it is used in typography. The *mean, descender* and *top/cap heights* are the same as they are defined in that field, as explained in Chapter 2.

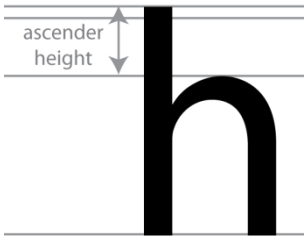

Figure 44. Ascender height for the application.

The localization of the *ascender line* and *top line* is going to vary depending of the size of the handwriting of each user although, in most cases, the *ascender* would be found above the *top line*.

For the first set of symbols and digits that are listed in Figure 45 row A and B, the height is measured as the distance between the *baseline* and the *ascender line*. In the row C symbols, the height is considered to be from the *descender line* to the *ascender line*. For the mathematical symbols in row D, the height is the space between the *baseline* and the *mean line*. Characters in row E have no defined height, since those particular symbols consist mainly of horizontal lines.

Row	Characters									
A	0	1	2	3	4	5	6	7	8	9
B	!	?	#	$	%	&	/	\		
C	@	[	]	{	}	(	)	|		
D	+	*	<	>						
E	-	=	_							

Figure 45. Character classification.

5.2 Partial Data from Metric Lines

Our approach has been to deduce metric lines from a collection of handwritten samples, and compute average values. Not all characters run across all the metric lines. Some do not rely on the *top line* or *descender line*. An example for this is the lowercase “h”, that has *ascender* and *mean height.* The character makes contact with *baseline, mean* and *ascender line.* Other examples are the lowercase “a” that only has *mean height,* and the letter “g” that has *mean* and *descender* height.

With the above, the average calculation is based or calculated with partial information. Partial data is extracted from the characters due to the fact that not all the symbols have the same metrics or all the available measures. This implies that not all of them have the same height.

For example, the characters that have an *ascender height* are the only ones taken in consideration to calculate the average measure of this height. The average descender height is measured in the same manner as the ascender height, by taking just those characters that have this height. For the *mean height* all characters are taken into consideration in calculating the average value.

5.3 Method

5.3.1 Determination of desired font size

The calculation of the output handwritten font size is made by computing the average size of each of the heights obtained from the acquired samples.

By obtaining these new measurements, the height of the handwritten output character can be adjusted making use of linear scaling and de-slant-scale-re-slant. If smoother transitions are desired, piecewise cubic scaling may be used.

5.3.2 Feature scaling

Scale

For the scaling process of the characters we use linear scaling, computed as:

$$new\ value = \ original\ value\ *\ scale$$

Formula A. Scale formula.

To describe the computations we use the following notation:

s_a = sample *ascender height*

y_a = desired new value of *ascender height* (average *ascender height*)

s_t = sample *top / cap height*

y_t = desired new value of *top / cap height*

s_x = sample *mean height*

y_x = desired new value of *mean height*

y_b = desired value of the *baseline*

s_d = sample *descender height*

y_d = desired new value of *descender height*

For each point (x, y) in the trace of each character, we calculate a new value ($x_{new,}$ y_{new}). The new values for the mean, ascender, top and descender height where calculated making use of the following formulas:

$$y_{new} = y_b - \left[\frac{y_b - y_x}{y_b - s_x} * (y_b - y)\right]$$

Formula B. New coordinates for y in the mean height region.

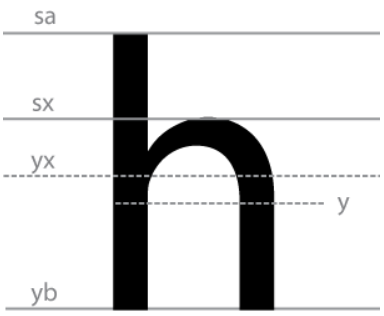

Figure 46. Variables from Formula B.

$$y_{new} = y_x - \left[\frac{y_x - y_a}{s_x - s_a} * (s_x - y)\right]$$

Formula C. New coordinates for y in the ascender height region.

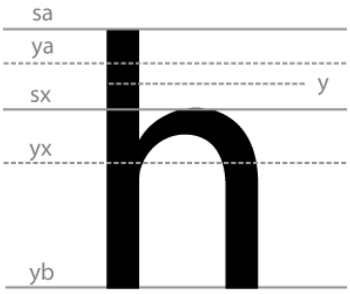

Figure 47. Variables from Formula C.

$$y_{new} = y_b - \left[\frac{y_b - y_t}{y_b - s_t} * (y_b - y)\right]$$

Formula D. New coordinates for y in the top height region.

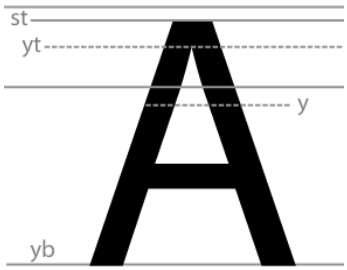

Figure 48. Variables from Formula D.

$$y_{new} = y_b + \left[\frac{y_d - y_b}{s_d - y_b} * (y - y_b)\right]$$

Formula E. New coordinates for y in the descender height region.

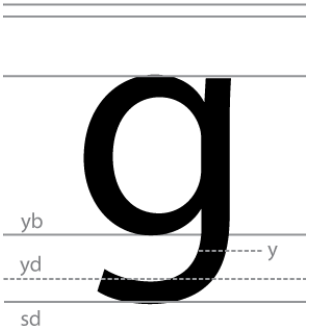

Figure 49. Variables from Formula E.

$$x_{new} = \left[\left(\frac{x_{newmax} - x_{min}}{x_{max} - x_{min}}\right) * (x_{current} - x_{min})\right] + x_{min}$$

Formula F. New value for x coordinates.

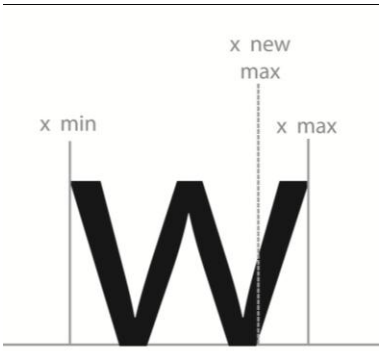

Figure 50. Variables from Formula F.

The slant on the characters could be easily overlooked, considering the method used for scaling is linear. However, with the following images a problem would be presented if the slant of the characters is ignored in the scaling process.

Figure 51. Original characters – example 1.

Figure 52. Original characters - example 2.

Figure 51 and Figure 52 present characters in two different handwriting fonts, with no alteration in height or width. Figure 53 and Figure 54 show the result of performing the neatening process in Figure 51 and Figure 52.

Figure 53 is the most notorious example to show that scaling the characters represents a problem when inclination is present. The ascender part of letter “b” was reduced to obtain the average ascender height, calculated with the value of all the ascender heights as mentions in Section 5.3.1. In Figure 54 the letter “g”, in both cases (left and right), presents the same alteration as letter “b” in the previous image. Although the distortions in the three characters have different intensity it is existent in all of the words.

To fix this problem we considered important to take into account the inclination of each character in the scaling process. A general inclination for the word is not considered since the characters are captured individually and may present different degrees of inclination.

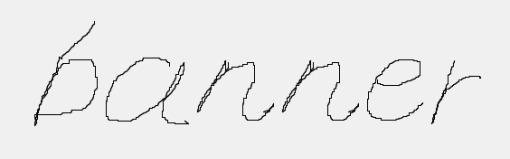

Figure 53. Neatened characters - slant ignored - example 1.

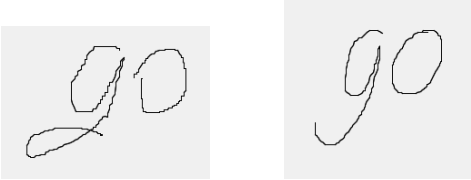

Figure 54. Neatened characters - slant ignored - example 2.

5.3.3 Conjugated feature scaling

De-slanting

To de-slant the characters, each font has its *x* coordinates shifted to the left or right by an amount *h*. We use the inclination of the character, θ, to calculate the value of *h*. The angle

(θ) is provided by the user, by adjusting the vertical line to the desired inclination, in the created tool, when the characters are being captured, as shown in Figure 56.

The new value of *x* will be obtained as follows:

$$x_{new} = \ x - h$$

Formula G. De-slant character.

Where:

$$h = \ \tan\theta \ * \ (y_b - y)$$

Formula H. Displacement for x coordinates in the de-slant of the characters.

The symbols take the following meaning:

h = distance that the *x* values need to be shifted to de-slant

θ = angle of inclination of the character

y_b = value of the *y* coordinate of the baseline

x = value of *x* coordinate of the current point

y = value of *y* coordinate of the current point

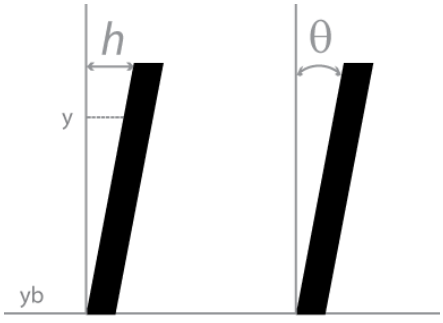

Figure 55. Variables from Formula H.

Re-slant

When the original handwritten characters have some inclination, it is necessary to return them to that original slant from the de-slanted state. For this purpose, re-slanting is performed. The formula and process is very similar to the one used for de-slant, explained before. The difference is in the value of *h*, that it is added instead of being subtracted.

$$x_{new} = x_{current} + h$$

Formula I. Re-slant character

Where *h* is calculated with

$$h = \tan\theta * (y_b - y)$$

Formula J. Displacement for x coordinates in the re-slant of the characters

Where

h = distance that the *x* values need to be shifted to re-slant

θ = angle of inclination of the character

y_b = value of the *y* coordinate of the baseline

y = value of *y* coordinate of the current point, to change in the character

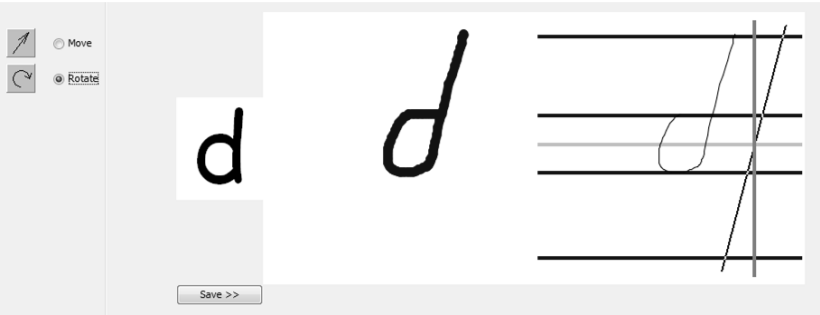

Figure 56. Capturing inclination of character.

In Figure 56 it is shown how the vertical line, in the created tool, is adjusted to have the same degree of inclination as the character. The value of the slant is stored along with the rest of the metric measures, to be used in the scaling process.

De-slanting of the characters is done before scaling them with the method explained in Section 5.3.3. Then the characters are slanted again. Thus, the scaling process is done on characters without the inclination, but the original slant is restored in the end. Figure 57 and Figure 58 show the result of this process, where the resultant characters have no distortion as in Figure 53 and Figure 54.

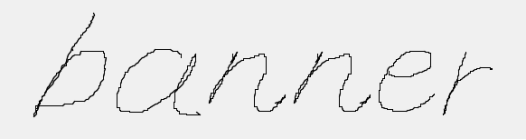

Figure 57. Neatened characters - considering slant - example 1.

Figure 58. Neatened characters - considering slant - example 2.

In Figure 59 image 1 is the original character. Image 2 is the scaled character when the slant is not taken into consideration and image 3 is the scaled character, where first the character was de-slanted, then scaled, and finally re-slanted to avoid distortions.

Image 2 shows how the ascender part of the character is affected when the character has an inclination and it is scaled before eliminating that slant. This is caused by the different proportions used to perform the scaling in the *ascender height* and in the *mean height*.

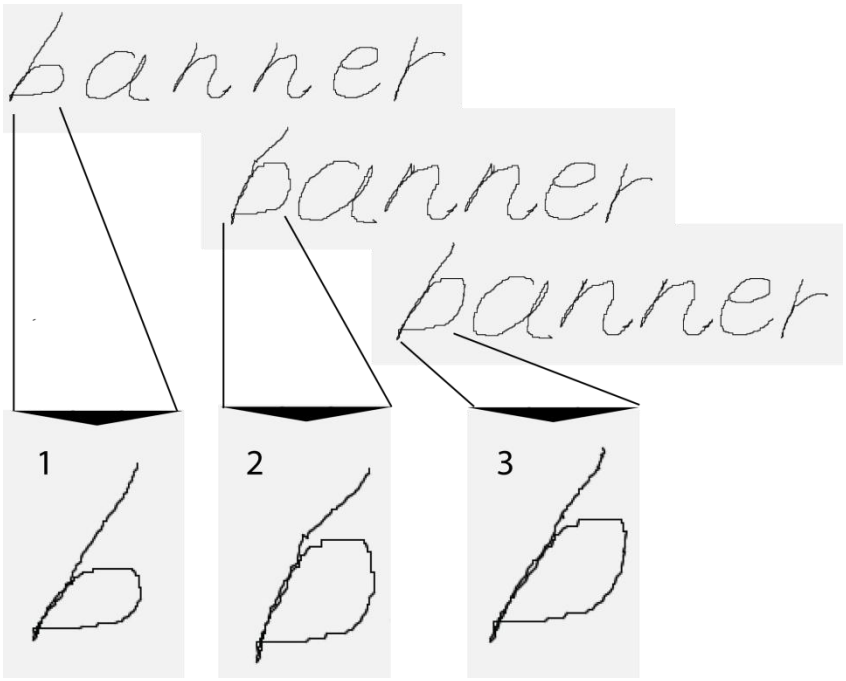

Figure 59. Character zoom in.

As mentioned above, the same situation happens with the *mean height* and *descender height* illustrated in Figure 54 and Figure 58

5.4 Personalized Text

The tool that we have created for annotating samples and for neatening characters accepts typed words as input. The information obtained in the process of collecting the handwritten samples of the characters is then used to perform the neatening process. The output consists of the typed word shown in one of the already existing or recently captured handwriting styles.

To visualize this better, the following images present the original captured samples and the output text.

Although the method used is linear scaling, which minimizes distortion; the resulting neatened handwriting characters might vary slightly from the actual handwriting style.

This may happen because of user's inability to capture the character and measure lines correctly. For example, the user might write the characters in varying sizes, therefore causing a very wide difference between widths and heights.

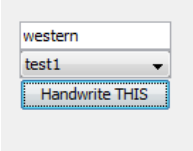

Figure 60. Example of input word.

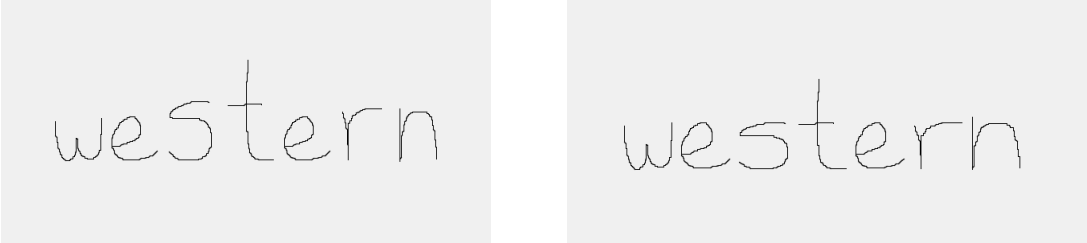

Figure 61. Example 1 of handwriting before and after neatening.

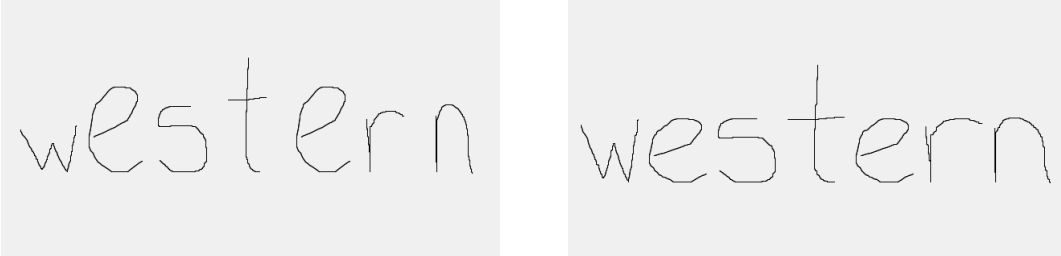

Figure 62. Example 2 of handwriting before and after neatening.

The first calculation for the new value of the width of the characters was based on the average width of all the letters that form the English alphabet. The results can be observed in Figure 61 and in Figure 62 . In the above images the characters have an average *mean height* (since all the letters in that example are lowercase) and a similar calculation is made for the width. The average width is calculated with all the captured characters taken

into consideration and not just the ones in lowercase like, as in the *mean height* calculation.

After further analysis of the results, it was seen that when we scale the characters to average width, it was altering the actual handwriting style. Hence, we decided to scale the width of the characters individually, using the same proportion as that used to scale the height of that character. This approach produced much better results, where the output preserved the original shape of the handwritten characters.

The following images, from Figure 63 to Figure 66, show the new obtained results from the neatening process of the characters. With the modified method to calculate the width of the characters, we have achieved a better image. Each example has a picture to illustrate how the word looks when no neatening is made on it and it is located on the left side of each example. The images placed on the right side are to illustrate the words after the neatening process.

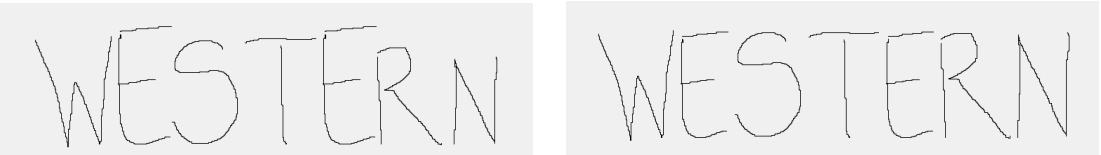

Figure 63. Example 3 of handwriting before and after neatening (uppercase).

In Figure 63 a word in uppercase letters is shown. In letters like the “S”, ”T”, ”R” and “N” the adjustments in heights are more obvious to the naked eye. Note the difference in the example in Figure 64 and Figure 65, where adjustments could be easily overlooked if there was no original image right next to the neatened one.

Figure 64. Example 4 of handwriting before and after neatening (lowercase).

Figure 65. Example 5 of handwriting before and after neatening (numeric).

The degree of change in the heights of the characters and their width will depend entirely on how the characters are captured and how accurate the metric lines are positioned, which means that it will vary from user to user.

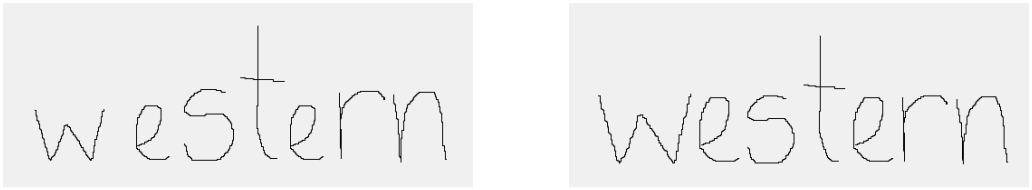

Figure 66. Example 6 of handwriting before and after neatening (lowercase).

When the characters are neatened, it is important to consider the slant in them. The figures above, show only examples of characters with no inclination.

CONCLUSION

We have examined the problem of neatening handwriting in the context of digital ink. We have found that it is possible to determine average sizes for ascenders, descenders, letter bodies, slant and other features by applying the concepts of font metrics from typography. This allows us to give a small set of numbers that characterize a writer's style. Then piece-wise scaling can be used to make newly written characters conform to this style. In order to preserve straight lines in the transformed characters it is necessary to conjugate the piece-wise transformations, by first de-slanting, performing the transformation, and then re-slanting to the desired slant.

We have developed a tool that allows us to capture font metrics from handwriting samples, and another tool that allows us to apply the transformations. The first tool is of general use for collecting metric information on samples and may be used for other purposes, for example to aid in handwriting recognition. The second tool is a demonstration of concept, and has a number of limitations.

We see this work as a first step in the work of handwriting neatening, and there are many further directions that could be pursued. These include:

1. Handling cursive writing with connected characters. This would involve using standard segmentation algorithms.
2. Automatic detection of the critical points for the font metrics.
3. Use of non-linear scaling methods.

4. Character specific metrics (for example, a user may always write a "b" higher than an "h").
5. Use of font metrics in base-line estimation for handwriting recognition systems.

This is just a part of what could be accomplished; handwriting neatening could present interesting challenges for years to come.

REFERENCES

[1] *LiveBoard: a large interactive display supporting group meetings, presentations, and remote collaboration.* **Scott Elrod, Richard Bruce, Rich Gold, David Goldberg, Frank Halasz, William Janssen, David Lee, Kim Mccall, Elin Pedersen, Ken Pier, John Tang, and Brent Welch.** New York, USA : Proceedings of the SIGCHI-Special Interest Group on Computer-Human Interaction- conference on Human factors in computing, 1992. pages 599–607.

[2] *InkBoard - Tablet PC Enabled Design Oriented Learning.* **Hai Ning, John R. Williams, Alexander H. Slocum, and Abel Sanchez.** Hawaii, USA : CATE -Computer and Advance Technology in Education-, 2004. pages 154–160.

[3] *Teaching with an Intelligent Electronic Chalkboard.* **Gerald Friedland, Lars Knipping, Raúl Rojas and Ernesto Tapia.** New York, USA : ACM-Association for Computing Machinery- SIGMM-Special Interest Group on Computer Human Interaction- workshop on Effective telepresence, 2004. pages 16–23.

[4] Handwriting Recognition Software for Pocket PC, Windows Mobile and Palm OS. *MobileWrite Handwriting Recognition.* [Online] Inkmark Software LLC, valid on June 2010. http://www.inkmarksoftware.com/index.html

[5] Handwriting Recognition - My Script. *My Script Handwriting Recognition.* [Online] Vision Objects 2009, valid on June 2010. http://www.visionobjects.com/handwriting-recognition/how-does-myscript-work/

[6] Ritescript Products. *Ritescript.* [Online] Evernote® Corporation 2009, valid on June 2010. http://www.ritescript.com

[7] Make a note. *Welcome to your notable world.* [Online] Evernote Corporation. Copyright 2010, valid on June 2010. http://www.evernote.com/

[8] Digital Ink, Breakthrough Technology in Tablet PC, Brings the Power of the Pen to the Desktop. October 29, 2002. *Microsoft News Center.* [Online] valid on June 2010. http://www.microsoft.com/presspass/features/2002/oct02/10-29TabletInking.mspx

[9] **Yao, Paul.** Tablet PC. MSDN Magazine 2004. *Add Support for Digital Ink to Your Windows Applications.* [Online] valid on June 2010. http://msdn.microsoft.com/en-us/magazine/cc163869.aspx

[10] Office OneNote 2007. *Microsoft Office Online.* [Online] Microsoft Corporation, valid on June 2010. http://office.microsoft.com/en-us/onenote/HA101656661033.aspx

[11] Real Cursive Handwriting Fonts. *Your Handwriting Font on Your Computer.* [Online] vLetter Inc. 2007-2010, valid on June 2010. http://www.vletter.com/index.htm

[12] Handwriting Font Services. *A World of Entertainment and Web Design.* [Online] Quantum Enterprises 2009, valid on June 2010. http://www.quantumenterprises.co.uk/fonts/index.htm

[13] Scanahand. *High-Logic Proven Font Technology.* [Online] 1997-2010. High-Logic B.V., valid on June 2010. http://www.high-logic.com/

[14] **Alexander Walter.** Handwriting Fonts. *Handwriting Fonts and Other Font Products.* [Online] valid on June 2010. http://www.walterware.com/

[15] YourFonts - Generate Your Own Fonts Online. *Font Generator - Make Your Own Handwriting Fonts With Your Fonts.* [Online] 2008-2010. YourFonts, valid on June 2010. http://www.yourfonts.com/

[16] *Handwriting Interface for Computer Algebra Systems.* **H. Okamura, T. Kanahori, M. Suzuki, R. Fukuda, W. Cong, F. Tamari.** Guangzhou, China : The Fourth Asian Technology Conference in Mathematics, 1999.

[17] *Local Slant Estimation for Handwritten English Words.* **Yimei Ding, Wataru Ohyama, Fumitaka Kimura.** s.l. : 9th Int'l Workshop on Frontiers in Handwriting Recognition - IEEE, 2004.

[18] Slate Corporation. *JOT - A Specification for an Ink Storage and Interchange Format.* [Online] valid on June 2010. http://unipen.nici.kun.nl/jot.html

[19] **Guyon, Isabella.** Unipen 1.0 Format Definition. *Data and Benchmarks for Handwriting Recognition.* [Online] Int Unipen Foundation. 1994, valid on June 2010. http://www.unipen.org/

[20] Windows Developer Center - MSDN Library. Microsoft Corporation. *Ink Data Format (Windows).* [Online] valid on June 2010. http://msdn.microsoft.com/en-us/library/ms702393%28VS.85%29.aspx

[21] **Yi-Min Chee, Katrin Franke, Max Froumentin, Sriganesh Madhvanath, Jose-Antonio Magaña, Gregory Russell, Giovanni Seni, Christopher Tremblay, Stephen M. Watt and Larry Yaeger.** Ink Markup Language (InkML). World Wide Web Consortium (W3C). *W3C Working Draft.* [Online] May 27, 2010. Valid on June, 2010. http://www.w3.org/TR/InkML/

[22] The W3C Multimodal Interaction Working Group. [Online] World Wide Web Consortium (W3C), valid on June 2010. http://www.w3.org/Consortium/

[23] **Xie, Xiaofang.** *On the Recognition of Handwritten Mathematical Symbols.* London, On. Canada : University of Western Ontario, 2007. Phd thesis.

[24] **James Craig, Irene Korol Scala, and William Bevington.** *Designing with type: the essential guide to typography.* New York, 2006 : Watson-Guptill, 1930.

[25] *A Framework for Pen-Based Mathematical Computing.* **Watt, Stephen M.** Beijing : Internet Accessible Mathematical Computation, July 24, 2005. IAMC .

[26] **Hui, Rui.** *Portable implementation of digital ink: Collaboration and calligraphy.* London, On. Canada : University of Western Ontario, 2009. MSc thesis.

[27] **Regmi, Amit.** *Supporting multimodal collaboration with digital ink and audio.* London, On. Canada : University of Western Ontario, 2009. MSc thesis.

[28] Typographic Terminology. *Web Design.* [Online] valid on June 2010. http://www.flywebmaster.com/webdesign/tips/type_terminology.php

[29] Typography Character Design Standards. *Typography - Developing Fonts.* [Online] (update 2009) valid on June 2010. http://www.microsoft.com/typography/developers/fdsspec

[30] **Turner, David.** FreeType Glyph Conventions. *The Free Type Project.* [Online] Copyright 1998-2000, valid on June 2010. http://www.freetype.org/freetype2/docs/glyphs/

CURRICULUM VITAE

Name:	María Teresa Infante Velázquez
Post-secondary Education and Degrees	The University of Western Ontario London, Ontario, Canada M. Sc. Computer Science September 2008 – June 2010.
	Universidad Tecnológica de México Mexico City, Mexico B. Computer Systems Engineering September 2001 - April 2006
Related Work Experience	Teaching Assistant The University of Western Ontario September 2008 - December 2009
	Research Assistant Ontario Research Centre for Computer Algebra September 2008 - April 2010