# Tablet PC Tool for Handwriting Recognition 

Nuno Resende

# U. PORTO <br> FEUP faculdade de engenharia UNIVERSIDADE DO PORTO 

Master's in Informatics and Computing Engineering
Supervisor: Prof. Alexandra Sofia Ferreira Mendes

# Tablet PC Tool for Handwriting Recognition 

Nuno Resende

Master's in Informatics and Computing Engineering

Approved in oral examination by the committee:

President: Prof. José Carlos Medeiros de Campos
External Examiner: Prof. Pedro Henrique e Figueiredo Quaresma de Almeida Supervisor: Prof. Alexandra Sofia Ferreira Mendes

## Resumo

Embora os sistemas de reconhecimento de caligrafia online tenham recebido um foco significativo recentemente, especialmente com a evolução dos algoritmos de machine-learning relacionados com o reconhecimento online, a estruturação e a manipulação da matemática tradicional, ainda há uma falta de ferramentas de software que considerem a forma do Calculational Method.

Software como o Windows OneNote Maths Assistant está significativamente avançado em relação ao reconhecimento, à interpretação, e até ao cálculo de soluções para problemas matemáticos. No entanto, é preciso haverem mais ferramentas específicas dedicadas aos utilizadores de provas de cálculo. A estrutura de provas de cálculo consiste numa declaração inicial a ser comprovada e uma série de passos que reescrevem ou reorganizam essa fórmula, aproximando-a do objetivo da prova iterativamente, fornecendo justificações para essas reorganizações entre cada passo. Uma razão pela qual estas provas são difíceis de incorporar em software pode estar relacionada com a notação incomum que é usada, principalmente porque essas provas podem exigir a introdução e a definição de novos símbolos, sobretudo servindo como operadores matemáticos com significados personalizados.

Inserir símbolos para fórmulas matemáticas num computador é geralmente difícil através de um rato e teclado, pelo que é preferível ter uma ferramenta que consiga reconhecer a caligrafia, algo mais natural para o ser humano. Devido aos avanços nas tecnologias baseadas em canetas, como o Tablet PC, escrever documentos matemáticos digitais tornou-se mais viável.

Portanto, esta tese descreve a criação de um sistema que liga uma interface de whiteboard a um reconhecedor existente - uma ferramenta que interpreta a matemática manuscrita tradicional - que será estendido para incorporar a estrutura e símbolos de provas de cálculo. Esta melhoria ajudará cientistas, investigadores e educadores a escreverem as suas provas e a verificá-las sem a sobrecarga da interação humano-computador através de inserçães com rato e teclado.

## Abstract

Even though online handwriting recognition systems have received significant focus recently, especially with the evolution of machine-learning algorithms related to the online recognition, structuring, and manipulation of traditional mathematics, there is still a lack of software tools that consider the Calculational Method's form.

Software like Windows OneNote Maths Assistant is significantly advanced regarding the recognition, interpretation, and even calculation of solutions for mathematical problems. However, there need to be more specific tools dedicated to users of calculational proofs. The structure of calculational proofs consists of an initial statement to be proven and a series of steps that rewrite or rearrange this formula, getting it closer to the proof's goal iteratively while providing justifications for those rearrangements between each step. A reason these proofs are hard to incorporate into software may be related to the uncommon notation used, mainly as these proofs may require introducing and defining new symbols, mainly serving as mathematical operators with custom meanings.

Inserting symbols for mathematical formulae in a computer is generally hard through a mouse and keyboard, which is why it is preferable to have a tool that can recognise handwriting, something more natural to the human being. Because of the improvements in pen-based technologies, like the Tablet PC, writing digital mathematical documents has become more feasible.

Therefore, this thesis describes the creation of a system that connects a whiteboard interface to an existing recogniser - a tool that interprets traditional handwritten mathematics - that shall be extended to incorporate the structure and symbols of calculational proofs. This improvement will support scientists, researchers and educators in writing their proofs and verifying them without the overhead of human-computer interaction through mouse and keyboard inputs.

Keywords: handwritten mathematics, calculational method, handwriting, online recognition
ACM Classification: Human-centered computing $\rightarrow$ Human computer interaction (HCI) $\rightarrow$ Interactive systems and tools

Human-centered computing $\rightarrow$ Human computer interaction (HCI) $\rightarrow$ Interaction devices

## Acknowledgements

I would like to thank Prof. Alexandra Mendes for the supervision and help on the development of this thesis.

I would also like to thank Dr. Francisco Alvaro and Dr. Ernesto Noya, who clarified some of my doubts and gave me insights related to the project's design.

I am very grateful to all the friends I made during my Erasmus, especially the Portuguese with whom I shared unforgettable experiences.

I sincerely thank all my friends who are part of the Amiguinhos community. You have made my life more joyful, and I hope to share many more fun moments with you.

A particularly heartfelt thank you goes to Kateřina, the woman I love. You will forever have a special place in my heart.

I would also like to express my deepest gratitude for the Bacalhaus, the friends who joined me in the journey of earning a degree. Çalves, Xavier, Gonçalves, Carvalho, Rafa, Romão and Coelho, you were all there in the darkest and brightest moments. I will never forget the countless laughs and drinks we shared.

Lastly, and most importantly, I would like to give special thanks to my parents, Darlindo and Paula Resende. I would definitely not be able to complete my Master's Degree without their unwavering support. I remain convinced that I have the best parents in the world. Muito obrigado por tudo que fazem por mim e pelo amor que me dão. Amo-vos muito!

Nuno Resende
"Do not try to fix whatever comes in your life. Fix yourself in such a way that whatever comes, you will be fine."

## Contents

1 Introduction ..... 1
1.1 Context ..... 1
1.2 Problem and Motivation ..... 2
1.3 Goals ..... 3
1.4 Document Structure ..... 3
2 State of the Art ..... 5
2.1 Background ..... 5
2.1.1 Offline Handwriting Recognition ..... 5
2.1.2 Online Handwriting Recognition ..... 6
2.1.3 Handwriting Recognition of Mathematical Expressions ..... 7
2.1.4 Calculational Method ..... 7
2.2 Related Work ..... 8
2.2.1 Natural Language ..... 8
2.2.2 Handwritten Mathematics ..... 10
2.3 Whiteboards ..... 13
2.3.1 OpenBoard ..... 13
2.3.2 Excalidraw ..... 13
2.3.3 Xournal++ Mobile ..... 14
2.3.4 Github Repositories ..... 14
2.4 Online Handwriting Recognisers of Mathematical Expressions ..... 15
2.4.1 Complete Systems ..... 15
2.4.2 Purely Recognitive Systems ..... 17
3 System Design ..... 20
3.1 Desiderata ..... 20
3.2 Handwriting-only vs. Handwriting and Keyboard ..... 21
3.3 Interface Design ..... 21
3.3.1 Generation of the N -Best Recognitions ..... 23
3.4 Recognised symbols ..... 23
3.5 Modular Design ..... 24
4 Solution and Implementation ..... 25
4.1 Architecture and Overview ..... 25
4.1.1 Client-Server Pattern ..... 25
4.1.2 Adapter Pattern ..... 27
4.2 Whiteboard ..... 28
4.2.1 InkML ..... 30
4.2.2 Display ..... 32
4.2.3 Erase Feature ..... 32
4.2.4 Download LATEX button ..... 33
4.2.5 Limitations and Assumptions ..... 33
4.3 Seshat ..... 34
4.3.1 Supported Symbols and Limitations ..... 36
4.4 ExpressJS Server ..... 38
4.4.1 Line Segmentation ..... 39
4.4.2 GET and POST handlers ..... 42
5 User Study and Validation ..... 43
5.1 Validation Goals ..... 43
5.2 User Study Instructions and Tasks ..... 44
5.3 Post-Experiment Questionnaire ..... 44
5.4 Methodology and Rationale ..... 46
5.4.1 Likert Scale ..... 47
5.5 Experimental Parameters ..... 48
5.6 Results ..... 48
5.6.1 Tasks ..... 50
5.6.2 Time ..... 56
5.6.3 System Usability Scale ..... 57
5.6.4 Spearman's Rank-Order Correlation ..... 58
5.6.5 Observations and Improvements ..... 60
5.7 Threats to Validity ..... 60
6 Conclusions and Future Work ..... 62
6.1 Summary ..... 62
6.2 Contributions ..... 63
6.3 Future Work ..... 63
References ..... 65
A User Study Guide ..... 70
B User Study Questionnaire ..... 74
C System Usability Scale Questionnaire Results ..... 84

## List of Figures

1.1 Equation in $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$ format ..... 2
2.1 Demonstration of different stroke orders ..... 6
2.2 Proof with Leibniz rule ..... 8
2.3 Proof that $\sqrt{2}$ is irrational ..... 9
2.4 Proof using rules of propositional logic ..... 9
2.5 Demonstration of Openboard ..... 14
2.6 Demonstration of MyScript Calculator ..... 16
2.7 Demonstration of MyScript Nebo ..... 17
2.8 Demonstration of Mathpix ..... 18
3.1 Demonstration of the current system's design ..... 22
3.2 Calculational Method's basic structure ..... 23
4.1 Sequence diagram of the system's architecture ..... 26
4.2 Adapter diagram representing the system ..... 28
4.3 Binary tree diagram of MathML ..... 32
4.4 Example of enclosed text ..... 33
4.5 Character "a" in different writing styles ..... 40
4.6 Diagram of table 4.1 ..... 41
4.7 Representation of the line segmentation calculation ..... 41
5.1 User study's proof 1 ..... 45
5.2 User study's proof 2 ..... 45
5.3 Ratings of digital literacy ..... 49
5.4 Familiarity level with handwriting recognition tools ..... 49
5.5 Most regularly used browsers ..... 50
5.6 Difficulty on writing proof 1 ..... 51
5.7 Difficulty on writing proof 2 ..... 51
5.8 Recognition accuracy on writing proof 1 ..... 52
5.9 Recognition accuracy on writing proof 2 ..... 53
5.10 Rewriting need on writing proof 1 ..... 53
5.11 Rewriting need on writing proof 2 ..... 54
5.12 Complexity of LATEX equivalent vs proposed tool on proof 1 ..... 55
5.13 Complexity of LATEX equivalent vs proposed tool on proof 2 ..... 55
5.14 Boxplot of time spent on proofs ..... 57

## List of Tables

4.1 Table of pen thickness vs vertical spacing ..... 40
5.1 Time spent in writing Proofs 1 and 2 ..... 56
5.2 Statistical analysis of proofs 1 and 2 ..... 56
5.3 SUS score for each participant ..... 58
5.4 Average SUS score for each question ..... 59
5.5 Spearman correlation and confidence levels ..... 60

## Abbreviations and Symbols

| (B)LSTM | (Bidirectional) Long Short Term Memory |
| :--- | :--- |
| CNN | Convolutional Neural Network |
| CROHME | Competition on Recognition of Online Handwritten Mathematical Expressions |
| CTC | Connectionist Temporal Classification |
| DTD | Dynamic Threshold Distance |
| GHA | Guided Hybrid Attention |
| GRU | Gated Recurrent Units |
| HME | Handwritten Mathematical Expression |
| HMM | Hidden Markov Model |
| OCR | Optical Character Recognition |
| RNN | Recurrent Neural Network |
| SCAN | Stroke Constrained Attention Network |
| SCFG | Stochastic Context-Free Grammar |
| SUS | System Usability Scale |
| SVM | Support Vector Machines |
| kNN | k-Nearest Neighbours |

## Chapter 1

## Introduction

1.1 Context ..... 1
1.2 Problem and Motivation ..... 2
1.3 Goals ..... 3
1.4 Document Structure ..... 3

This chapter introduces the challenge at hand and the impetus for addressing it. Section 1.1 discusses the context around handwriting recognition. Section 1.2, p. 2 defines the problem in light of the context presented previously and explains the motivation to solve this issue. Section 1.3, p. 3 mentions the project's goals. Finally, Section 1.4, p. 3 provides a roadmap for the remainder of the document.

### 1.1 Context

Performing tasks with our hands is part of what has defined humans throughout evolution, especially in creating tools and utensils. This is likely the main reason humanity is the world's dominating species. So, understandably, we tend to find handwriting as a natural form of inputting representations of our perception of reality, whether by sketching or writing.

Mathematics, in particular, is deeply bound to how it is written, given that we attribute meaning to the symbols used depending on their position. As an example, let us take a look at the following mathematical expression:

$$
\begin{equation*}
\int_{0}^{10} 5 \times \frac{3 x^{4}}{2} \tag{1.1}
\end{equation*}
$$

Observing it closely, it can be easily understood how the two-dimensionality gives each symbol a clear, distinct importance and how writing formulae in a single, straight line would be unfit for our necessities in depicting a mathematical idea. This spatial importance is also seen when considering the spacing of numbers with multiple digits or operators typically represented with natural language characters like cos.

$$
\text { \int_\{0\}^\{10\}5\times\frac\{3x^\{4\}\}\{2\}\}}
$$

Figure 1.1: Equivalent of the equation 1.1 shown written in the $\mathrm{LT}_{\mathrm{E}} \mathrm{X}$ format

On a computer, traditionally, input by the user is made through peripherals like the keyboard and mouse, as their combination allows for efficient human-computer interaction. However, when dealing with mathematics, to produce the Equation 1.1 presented in the ${ }^{\mathrm{ET}_{\mathrm{E}} \mathrm{X}}$ format, for example, one would need to type the expression in a complex way, as shown in Figure 1.1.

It becomes clear that introducing such a formula with the keyboard needs to be more intuitive and legible. It provokes an efficiency overhead, especially considering the necessity of inputting special symbols like Greek characters in the form of the commands that represent them.

Since writing is natural and adequate for mathematical representation, introducing systems for handwriting recognition is a logical consequence and has many applications, for instance, for educational purposes. In the current world where digital technology has become an integral part of our lives, education must become a subject of the same adaptation and transition. This technological shift has allowed for new methods of teaching to be developed that are independent of traditional physical constraints, providing access to more people, and mathematics education is at the core of the migration from learning environments in a classroom to a digital one.

However, this shift is not only beneficial for educators. There are new opportunities for mathematicians, scientists and researchers to propagate their ideas and collaborate. Digital whiteboards offer the opportunity to replicate the experience of jotting down mathematical proofs in the calculational style that can be recognised and validated in the future.

### 1.2 Problem and Motivation

This thesis arises partly in the context of the Mendes' work [43, 44] that had the goal of being an assistant that teaches the dynamics of algebraic calculations and syntactic manipulation, specifically by providing the possibility of selecting, copying, and applying algebraic rules on a Tablet PC, manipulating the formulae's structure with gestures. Math $\int$ pad [6], a structure editor for mathematical documents that assists in the process of doing mathematics, was an essential inspiration for the author. The end product is a modular library implemented in C\# integrated into Classroom Presenter, a system for presentations and classroom interaction whose results proved it suitable for its intended purpose. They also concluded that most of the recognisers available at the time were proprietary software, and none seemed to support all desired symbols and constructions, which is still valid. For that reason, the system uses a recogniser provided by Maplesoft, which is limited to recognising only individual symbols, not considering the context in which they are used, leaving their focus on improving structure analysis and editing. This thesis aims to fill that gap with a more suitable recogniser. In order to achieve optimal outcomes, this dissertation's recogniser should also incorporate the context surrounding the written symbol. For instance, if a given user has written the expression " $3+$ ", the recogniser should infer that the next symbol is
more likely to be a number. Similarly, in the case of the subsequent input being $x$, for example, the recogniser should also consider that symbol more likely to represent the lower-case character " $X$ " rather than the multiplication operator $\times$. By considering the context in which the symbols are written, the recogniser can perform better and make more informed interpretations of the intended meaning.

Despite their advantages, recogniser tools present a unique set of challenges when it comes to recognising handwritten mathematical expressions and proofs, particularly in the calculational style. Existing software does not effectively support the fluid nature of mathematical ideation, as users frequently need to pause their thought processes to ensure that their handwriting fits the constraints of the recognition systems. These interruptions can hinder the cognitive flow, removing the user from an immersive experience. Moreover, current tools need to be tailored to capture the complexity and intricacy of these proofs that contain unconventional syntax. Current recognition systems fail to accurately interpret notations of the Calculational Method, struggling to understand the semantic relationships between symbols crucial for correctly interpreting mathematical proofs.

The problem mentioned forms the driving force behind the present thesis. There is a clear gap in existing research and tools concerning the online handwriting recognition of mathematical proofs in the calculational style, which becomes an even more pressing issue in the face of the world's digital transformation. Developing a tool tailored to the recognition of proofs in the calculational style can streamline the process of sharing, and validating mathematical proofs, increasing productivity.

### 1.3 Goals

The main goal of the thesis is to tackle the need for more software for people who want to bring handwritten annotations of their proofs into a digital format. Essentially, what needs to be addressed is the development of a tool that can accurately and efficiently recognise online handwriting of mathematical proofs in the calculational style on a digital whiteboard, accommodating different individual handwriting styles. Therefore, the project aims to develop a tool that combines a whiteboard frontend interface with a backend recogniser for handwritten mathematics that can be extended to recognise proofs of the Calculational Method accurately. The whiteboard interface and the recogniser should be connected to a server that handles requests sent by the whiteboard, processing the data and sending the final recognition back to the whiteboard for display.

### 1.4 Document Structure

This thesis contains four additional chapters. Chapter 2, p. 5 describes the state of the art and details the advances of related work and their existing challenges. Chapter 3, p. 20 properly focuses on the rationale behind the system's design choices and the project requirements. The implemented solution is explained in Chapter 4, p. 25. In an attempt to validate the solution, Chapter 5,
p. 43 reports the user study and experiments performed. Finally, Chapter 6, p. 62 presents the thesis' conclusions and future work.

## Chapter 2

## State of the Art

2.1 Background ..... 5
2.2 Related Work ..... 8
2.3 Whiteboards ..... 13
2.4 Online Handwriting Recognisers of Mathematical Expressions ..... 15

This chapter details the significant evolutions and related work relative to online handwriting recognition, starting by clarifying some core concepts that are essential for the dissertation's comprehension on Section 2.1, then briefly analysing natural language advances and focusing on handwritten mathematical expressions afterwards on Section 2.2, p. 8. Section 2.3, p. 13 and Section 2.4 , p. 15 show the existing relevant whiteboards that can be used and extended as the interface, and also existing online handwriting recognisers.

### 2.1 Background

With the development of handwritten recognition over the last decades after its groundwork around 1970 [3], researchers have come to distinguish handwritten recognition in two forms: offline and online recognition [61]. Since this thesis addresses the need for further investigation on (online) handwritten recognition of mathematical proofs in the calculational style, it becomes clear that all of these core concepts must be thoroughly clarified so that the rest of the dissertation is comprehensible.

### 2.1.1 Offline Handwriting Recognition

Offline handwriting recognition is a significant area of research in the fields of pattern recognition, computer vision and machine learning that refers to the identification of handwritten data in the format of real ink on a tangible, physical medium like paper by having the document scanned or analysing any static image containing a manuscript. It usually consists of the acquisition of data, followed by segmentation into words or characters, optical character recognition (OCR) [47] methods and post-processing [33, 36, 37, 64].


Figure 2.1: Demonstration of the number 5 in written form and with different stroke order labels

The process usually starts with image pre-processing to improve image quality so that system can better identify the text. This text then goes through a segmentation phase, where the system attempts to divide the written text into smaller units, such as words or characters. Once the content is pre-processed and segmented, feature extraction methods are performed, which may look into pixel density or region densities, for example. After all this processing, the recognition phase is typically carried out using algorithms trained on a dataset of handwritten images and their corresponding text labels. Lastly, the post-processing stage corrects errors and improves recognition accuracy by using dictionaries or language models to fix spelling and grammar mistakes.

### 2.1.2 Online Handwriting Recognition

Although both methods are used for recognising natural languages and mathematics, this thesis focuses on online recognition of mathematics, explicitly targeting the non-traditional mathematics used in calculational mathematics and proofs.

Online handwriting recognition is related to real-time recognition, meaning that the recognition is as instantaneous as possible and happens while the user writes with a special pen on an electronic device. It is essentially the tracking of information through the touch of this pen on the surface of a device, monitoring the $x$ and $y$ coordinates of the pen with the help of a sensor that gathers the up-and-down shifting, ultimately transforming the writing as quickly as possible into interpretable symbols and expressions. This technology does not process static images of handwriting. Instead, it considers dynamic information like the sequencing and velocity of the stroke. It also considers temporality, keeping information of when and potentially how each character is written, particularly the direction of the strokes, the applied pressure, speed, and others. This information brings an advantage over offline recognition as this data can be used to disambiguate similar-looking characters and improve accuracy on the recognition [9]. Nevertheless, each person's strokes are ordered differently in some symbols, potentially turning the recognition into a more complex task. Figure 2.1 shows this using the number five as an example, having the top horizontal line drawn first by some people and second by others.

Furthermore, every person has their own handwriting style, meaning the characters may be drawn more or less cursively and with more or less space between them, creating issues related to segmenting the expression for later analysis. The symbol identification process is further complicated if the characters are similar by nature, for example, in the case of "V" and "U". For this
reason, there is usually the application of some pre-processing first, and only afterwards is the word adequately structured. These issues also apply to recognising common symbols in mathematical notation, for example, in the case of "C" and "(" [12]. In the case of this thesis, both should be addressed.

### 2.1.3 Handwriting Recognition of Mathematical Expressions

Handwriting recognition of mathematical expressions refers to a subset of handwriting recognition technology that, instead of focusing on deciphering written text characters or numerals, tries to interpret complex mathematical symbols and structures, ranging from simple equations to intricate formulae. It is a highly complex and challenging study area due to its characteristic problems. The diversity of mathematical symbols is considerably more significant than the alphabets, with symbols varying across different mathematical disciplines. These systems must not only correctly categorise handwritten symbols but also understand the rules of mathematical notation to interpret the structure and semantics of written expressions with precision. These systems trained on large datasets of handwritten mathematical expressions often employ a combination of pattern recognition techniques and processing to accomplish these tasks.

An additional problem with handwriting recognition of mathematics is that it is nearly impossible to establish a dataset that acts as the equivalent of a dictionary in natural language recognition. Creating storages of words enables us to match the writing with that previously collected knowledge, allowing for a performance boost. However, that is hard to achieve in mathematics. While it is possible to store the users' strokes and have the tool learn their handwriting style, users can also write any possible combination and variation of symbols.

### 2.1.4 Calculational Method

Mathematical proofs have helped mathematicians and scientists get a new perspective on some problems for over two thousand years. Research on correct-by-construction elevated the current state of proofs based on logic and propositional calculus [23, 31] by conceiving a calculational style [5, 62] that emphasises the use of calculations and transformations to derive results and substitutes verbose proofs with clear calculation steps presented in a uniform format. This way, creating mathematical proofs has become more accessible and has enhanced the overall understanding and application of mathematical concepts. The Calculational Method considers the importance of precision and conciseness, employing symbolic and computational expressions instead of purely textual explanations. In this style, the arguments are stated through mathematical formulae that are manipulated and complemented with hints in natural language, allowing for ease in comprehending each step's transition and elucidating the logic and principles that guide these transitions. Naturally, a given proof with $n$ statements, shall have $n-1$ hints. By proceeding in this manner, one can focus on the correct application of logical rules first, taking an interest in the meaning of the symbols and interpreting the results in the later stages [43, 44]. In essence, the Calculational Method encourages a proof style highlighting the rules and operations, making the process more

```
    23\times11=243
{ { Leibniz (substitution of equals for equals) }
        (23\times11) mod 3 = 243 mod 3
= { (23\times11) mod 3 = 1
    243 mod 3 = 0
                        (details of calculation omitted) }
        1=0
= { arithmetic }
    false .
```

Figure 2.2: Example of a proof in the calculational style using the Leibniz rule and algebraic manipulation
systematic and analysable. The Calculational Method is a powerful application in mathematics and is important in computer science and software engineering, especially in formal verification and algorithm design. Its methodologies resonate strongly with software engineering and algorithm development principles, making it a practical approach in these domains.

Figure 2.2, Figure 2.3, p. 9 and Figure 2.4, p. 9 show different kinds of proofs with the structure of calculational mathematics. Although the content and deep understanding of these proofs are not crucial for this section, it is helpful to note that they have different styles, particularly in the nature of the hints provided. Figure 2.2 shows a proof with little text, a rule of propositional logic, Leibniz, and traditional mathematics, the proof on Figure 2.3, p. 9 uses a lot of textual description, and the proof on Figure 2.4, p. 9 shows some classic rules of calculational logic.

### 2.2 Related Work

The analysis of the related work on online handwriting recognition involved compiling some notable studies and papers that highlighted and documented the most important scientific breakthroughs, technological developments, and new approaches in online handwriting recognition at their respective times. These works sought a thorough analysis of the advancements achieved by detailing innovations and methods that have influenced and improved the technology surrounding handwriting recognition. They also pursued improving the understanding of the development of handwriting recognition by exploring the challenges involved and clarifying the tradeoffs of each method. Each one of these papers is going to be analysed individually.

### 2.2.1 Natural Language

Natural language recognition encompasses all 7000 languages [39], each with its peculiarities. Different languages work with fundamentally different schemes and systems, some having predefined symbols placed in sequence as in the Alphabetic system, and others having just combinations of lines and strokes in specific orders, allowing for thousands of meanings, which is predominant


Figure 2.3: Example of a complete proof in the calculational style showing $\sqrt{2}$ to be irrational with particularly descriptive hints

```
    \(p \wedge q\)
\(=\{\) golden rule \}
    \(p \equiv q \equiv p \vee q\)
\(=\quad\{\quad\) equivalence and disjunction are symmetric \(\}\)
    \(q \equiv p \equiv q \vee p\)
\(=\quad\{\quad\) golden rule, \(p, q:=q, p\) \}
    \(q \wedge p\).
```

Figure 2.4: Example of a proof in the calculational style using rules of propositional logic
in Asian languages. However, one may argue that the structure of natural language is more accessible to process than handwritten mathematical expressions (HMEs), as the writing is usually linear in the $Y$ axis.

### 2.2.1.1 Advances in online handwritten recognition in the last decades (2022)

Ghosh et al. [28] describe in their paper the latest advances of the decade in online handwriting recognition of natural language. The recognition can be broken down into four methods: stroke-based recognition, where a single character is used as the fundamental unit, characterbased recognition, which is used to segment a character or word into its component characters, using a recognition technique to recognize each letter afterwards, word-based recognition, where words are presented as a sequential combination of letters requiring a dictionary for vocabulary recognition, and line-based recognition methods, where the written text is separated into lines and the recognition is made of each line separately.

For stroke-based recognition, they describe systems of scripts for Gujatari, Assamese, Chinese, Kanji, Devanagari, Telugu, Tamil and Arabic using methods like, Hidden Markov Models (HMM) [55], Stochastic Context-Free Grammar (SCFG) [17], Support Vector Machines (SVM) [20], k-Nearest Neighbor (kNN) [21], and Multi-layer Perceptron (MLP) [57]. Some of the systems proposed have drawbacks such as high similarity of characters, identical characters and confusing characters, potentially affecting the accuracy of recognition.

In character-based recognition, they highlight research that focuses on addressing the issue of similar characters, producing promising results. Various techniques are used for Tifinagh, Arabic, Devanagari, Tamil, and Bengali, with recognition rates varying depending on the approach used, with some methods achieving high accuracy rates. However, some methods may be slower or have a more complex architecture.

Regarding word-based recognition, they present methods and techniques such as HMM, SVM, and Neural Networks [42] applied to languages like Arabic, Tamil, Bangla, Chinese, and Persian.

Lastly, for line-based recognition, they report methods such as kNN, Decision Trees [54], Neural Networks, and SVM for Arabic, Cyrillic, Devanagari, Han, Hebrew and Romanian.

Other works based on multi-language recognition are briefly discussed. This includes a framework using a multi-language database of 46000 words in Latin, Arabic, and digits and another system for recognising 102 languages. The authors also suggest that these studies need more research in this area in regard to constrained databases, multi-script documents, and confusing characters.

### 2.2.2 Handwritten Mathematics

Within the broad field of pattern recognition and machine learning, the recognition of handwritten mathematics is a particularly intricate and complex area of study. Since handwritten notation is still a persistent form of communication, particularly in educational settings, and mathematics is still the uniting language of science and engineering, solving its unique challenges is essential.

These challenges are enlarged by the complexity of reading mathematical symbols and comprehending their relationships, which are significantly different from the challenge of reading regular handwriting or even typeset mathematics. Handwritten mathematical writing typically employs a two-dimensionality with fractions, superscripts, and subscripts in addition to various symbols and operations. The inherent unpredictability in how various people write certain symbols and the semantic structure of mathematics makes the process more demanding.

Over the decades, several sophisticated strategies have been developed to deal with these issues, with advancements in preprocessing methods, feature extraction methods, classification algorithms, symbol segmentation, and structural analysis to achieve practical applications. Additionally, machine learning, particularly deep learning, is often used. These methods extensively use datasets of HMEs to develop models that effectively understand the patterns and structures of such notations.

Despite the progress that has been accomplished, some difficulties remain. The various mathematical symbols, expression formats, and the unpredictable nature of handwriting are only a few of the ongoing challenges. The boundaries of practicality for HMEs are being pushed by continuing research and technological advancements. In the following paragraphs, some of those are detailed.

### 2.2.2.1 Syntax-directed recognition of hand-printed two-dimensional mathematics (1967)

Research on the recognition of HMEs started being under the lens with Anderson's work [3], describing the problem of recognising mathematical expressions and matrices, which differs from previous research in the sense that the hand-drawn characters were not connected. Even though the practical, commercial applications were yet to be in sight, two applications were shown, a syntax for recognising arithmetic and one for recognising descriptions of matrices. They present a collection of rules to handle hand-printed characters using a top-down, syntax-directed recognition algorithm implemented in LISP 1.5, meaning that syntax rules direct the choices of sub-goals read as parametric data. The paper concludes that further research is needed to overcome slow performance and error detection.

### 2.2.2.2 Mathematical expression recognition: a survey (2000)

Later developments in the recognition of HMEs in the late twentieth century focused on comparing different systems in each stage of the recognition process, which they thought to be symbol recognition and structural analysis [15]. Symbol recognition consists of two substeps: segmentation, where the pen strokes are joined in a group, and classification, where the symbols are effectively identified. Related work of global solutions that treat both simultaneously will also be discussed.

Character recognition has existed for five decades, and yet the task is not trivial, partly because the operators and expressions have particular properties, such as context-sensitive components and explicit or implicit operators, which leads to having to take care of each symbol's peculiarities individually, threatening to be inefficient. The author also states that several techniques for symbol
recognition assume that the symbols have already been isolated from each other and therefore, prior segmentation is required.

When it comes to segmentation, Chan et al. show three works. The first one uses relation trees [25], deciding how to perform the segmentation based on projections on the X and Y axes. The second is a partition of symbols in components through profile cutting [51]. Lastly, a progressive grouping algorithm based on confidence levels is presented [58]. An issue with these methods is that they rely on thresholds, but these cannot be chosen perfectly to be compatible with all imaginable inputs. For example, choosing the stroke width threshold while segmenting into individual characters, distinguishing between actual strokes and noise, is not trivial because, depending on its value, the segmentation can result in missing strokes or splitting a stroke into multiple segments.

Different approaches like template matching, structural, and statistical approaches are described for recognition. Statistical methods like SVM [60, 7] and MLP [27] are commonly used as they offer high accuracy. It is also possible to have the segmentation and the recognition be performed simultaneously, for example, by using HMMs. This is a rather logical idea, as HMMs are helpful when dealing with continuous signals in time, which happens in the processing of speech signals and real-time strokes. The survey states some problems, however, like the possible ambiguity in the expressions [45] - mitigated if context information is used - lack of error detection, few considerations on the practical evaluation of the system's performance, and low accuracy. The authors also mention future work in applications such as handwriting interfaces for computer algebra systems, pen-based programs and mathematical tutoring systems [43, 44].

### 2.2.2.3 Application of deep learning in handwritten mathematical expressions recognition (2020)

In more recent work, the current standing of online recognition of HMEs were surveyed, particularly exploring the deep learning techniques and their differences towards the previously mentioned traditional solutions [41].

Regarding segmentation, Alvaro et al. [1] proposed a method of detecting components of a symbol by performing linear interpolation and later grouping them. AdaBoost [26] was used for segmenting with confidence weighted predictions by computing shape context features and then classifying the symbol scores for the strokes [34]. However, as mentioned, the segmentation and recognition of expressions do not have to be separate sequential processes. Awal et al. [4] suggested a system in which simultaneous optimization of segmentation, symbol recognition, and 2D structure recognition HMEs occurs, a model that learns the spatial relations of handwritten expressions. Encoder-decoder end-to-end framework ${ }^{1}$ were also introduced [67] with an attention mechanism using a stack of bi-directional recurrent neural networks (RNN) [24] and a parser using gated recurrent units (GRU) [16], further improved by adding guided hybrid attention (GHA) [68]. Encoder-decoder frameworks have been helpful in other applications like machine translation [63] and pen trajectory recovery [11]. Later, they also applied a tree-based long short term memory

[^0](LSTM) [30] solution by considering temporal and spatial relations in the input strokes [69] and, most recently, a stroke-constrained attention network (SCAN) that make use of stroke-level information instead of considering trace points as the basic units for recognition. It is important to note that all of these global solutions come at a computational cost, as the possibilities for segmentations should be maintained for a decision to be made only at the highest level and that some missing global information can lead to the incorrect recognition of delayed strokes. Convolutional neural networks (CNN) [38] are adequate for processing spatial information of two-dimensional data, which exists in online handwriting recognition [22], but has its primary use in offline processes like image classification and audio retrieval. RNN are commonly used because they model and process sequences - handwritten mathematical expressions are sequences [48] -, but they are prone to the vanishing gradient problem [32,8] with long sequences. Like in the work of Zhang et al. [68], a potential solution lies in using LSTM networks, where a purposefully built memory cell is included. If future and past information is required, one should use the bidirectional LSTMs.

### 2.3 Whiteboards

Regarding the choice of a whiteboard, almost any whiteboard can be extended into an interface for this project, as no particular features are being looked for beyond the ability to receive digital ink input and edit the writing slightly. Therefore, some whiteboards will be mentioned, all of which are appropriate candidates.

### 2.3.1 OpenBoard

OpenBoard ${ }^{2}$ is an open-source cross-platform whiteboard designed to be used in schools and universities since it can be used both with interactive whiteboards or in a dual-screen environment with a pen-tablet display and a beamer to create dynamic presentations, annotate documents and engage in real-time collaboration with others. Figure 2.5, p. 14 demonstrates how OpenBoard works.

### 2.3.2 Excalidraw

Excalidraw ${ }^{3}$ is an open-source, web-based collaborative whiteboard for sketching hand-drawn diagrams, editing and commenting on drawings. They provide a wide range of customisable drawing tools, pre-designed templates, drag-and-drop features for dealing with objects in the canvas and the ability to embed it into other websites.

[^1]

Figure 2.5: Demonstration of the interface of OpenBoard with the equation $\frac{x^{2}}{4}+1=y$ written

### 2.3.3 Xournal++ Mobile

Xournal++ Mobile is an open-source port of Xournal++ ${ }^{4}$, also an open-source software, developed for Flutter and can render strokes, images, text, and $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ and PDFs while including saving and basic editing features. However, some known issues exist, like considerable memory consumption upon opening large files.

This software was given more consideration than, Xournal++, also an open-source crossplatform handwriting notetaking software, because, although it contains a lot of exciting features like pen pressure support, tools for drawing shapes like lines and arrows, choosing between different paper types and a customisable interface, it was deemed overly complex considering nature of the features that needed to be added for the project. Additionally, as it is written in C++, adding new features requires more effort from a developer's standpoint simply due to the language's complexity when compared to a language like JavaScript.

### 2.3.4 Github Repositories

In GitHub, we can also find repositories holding whiteboard projects likely to be open-source, approachable and accessible. Although there are a myriad of different repositories available, we shall only describe two of them for the sake of efficiency.

### 2.3.4.1 XBoard

XBoard ${ }^{5}$ is a web-based whiteboard tool in Python that enables users to collaborate and write together in real-time, providing ways to annotate and manipulate images.

[^2]
### 2.3.4.2 cracker0dks/whiteboard

A repository called "whiteboard" is maintained by the user "cracker0dks" which hosts a collaborative whiteboard written in NodeJS that is easily customisable and presents similar features to XBoard, being a sketchboard that gives options of interacting with images and PDF files, writing text and placing sticky notes. It also includes features to move the canvas and drag-and-drop images ${ }^{6}$.

Ultimately, this last tool was chosen for its simplicity, which proves advantageous not only from the user's perspective, thanks to its intuitive and straightforward interface that incorporates all desired features for a handwriting frontend interface, but also from a developer standpoint as the codebase is concise and easy to comprehend.

### 2.4 Online Handwriting Recognisers of Mathematical Expressions

Although multiple recognisers and applications with various features related to handwriting recognition and manipulation exist, in this section, we shall only look at existing recognisers that also support the writing of mathematics in any form. The first subsection concentrates on systems with completeness in the sense that they provide user interfaces for handwriting and backend recognition. These applications are a single bundle, incorporating all demands of a typical user. The second subsection mentions software that does not include frontend interfaces but delivers recognition.

### 2.4.1 Complete Systems

### 2.4.1.1 MyScript Calculator

MyScript is a company specialising in handwriting recognition and digital ink management technologies. MyScript's Calculator ${ }^{7}$, a mobile app, uses handwriting recognition algorithms to translate handwritten equations into a computer-readable format by writing on the device's touchscreen. MyScript Calculator is widely used for general educational purposes, solving mathematical problems and calculating operations including simple arithmetic, trignometry, etc. Figure 2.6, p. 16 shows an example of its usage.

### 2.4.1.2 MyScript Nebo

MyScript is also the creator of Nebo ${ }^{8}$, a note-taking app designed to be used with a stylus that lets users write and organise their notes in a digital format. It mainly focuses on recognising and interpreting handwriting, converting handwritten notes into editable text and allowing for the formatting of those notes, and adding diagrams or extra sketches and flowcharts. While it is primarily a note-taking app, it is particularly well-known for its ability to recognise and solve traditional

[^3]$$
\frac{x^{2}}{4}+1=y
$$

## $\odot 0$



Figure 2.6: Demonstration of the interface of MyScript Calculator with the equation $\frac{x^{2}}{4}+1=y$ written
mathematical equations, which sets it apart from many other note-taking apps. Even though it may not provide real-time results or solve complex equations like MyScript Calculator, it captures handwritten math equations and works with them as digital text. Figure 2.7, p. 17 demonstrates Nebo's usage. As we can see, the recognition is not flawless since $X$ is being recognised as a multiplication operator and the fraction being detected as a subtraction. This example showcases the complexity of the handwriting recognition of mathematical expressions that even commercial, licensed, paid products cannot address without errors.

### 2.4.1.3 EquatIO

EquatIO ${ }^{9}$, a cross-platform tool by TextHelp, provides an online recognition tool that allows users to create mathematical expressions using a whiteboard. It uses advanced OCR technology to recognise and process the into a digital format. The software also includes voice dictation, reconnising a wide range of mathematical symbols, making it an effective tool for creating complex equations. EquatIO can be integrated with learning management systems, and it provides add-ons for platforms like Microsoft Word ${ }^{10}$ or Google Docs ${ }^{11}$. It also includes features of automatic completion, predicting and suggesting what the user might want to write next, increasing productivity.

Unfortunately, all of the recogniser systems mentioned so far are under commercial licenses. They are, therefore, not suitable to use in this thesis, as the objective of this work is to provide an open-source project that any scientist or educator can extend to accommodate their specific needs.

[^4]

Figure 2.7: Demonstration of the interface of MyScript Nebo on a mobile phone with the equation $\frac{x^{2}}{4}+1=y$ written

### 2.4.2 Purely Recognitive Systems

### 2.4.2.1 Mathpix

Mathpix ${ }^{12}$ is primarily an offline recogniser that uses OCR to interpret and convert handwritten equations and formulas into $\mathrm{LAT}_{\mathrm{E}} \mathrm{XT}$ They allow users to input mathematical expressions by taking pictures with a smartphone camera or uploading an image, transforming the expression into a digital format that can be easily edited and exported. However, they also include online recognition as they maintain an API that allows for integration into tools like the kind we want to achieve. In this API, the client may send a POST request containing the strokes data, and upon success, they will receive the $\mathrm{LT}_{\mathrm{E}} \mathrm{X}$ of the recognised handwriting with each line separated by a newline character. This API would be a great contender for the tool we would extend if it were not for the fact that it is a paid, closed-source service. Figure 2.8 , p. 18 shows an example of the application's normal usage without the API.

### 2.4.2.2 WritePad SDK

Regarding open-source options, PhatWare, a company specialising in handwriting recognition and digital ink technologies, provided a software development kit that allows users to integrate handwriting recognition into their applications using APIs ${ }^{13}$. WritePad can be used on multiple

[^5]

Figure 2.8: Demonstration of the interface of Mathpix with the equation $\frac{x^{2}}{4}+1=y$ written
platforms and recognises text in some of the world's most predominant languages, allowing developers to integrate handwriting recognition across different devices. It contains gesture recognition capabilities, so users can delete words by crossing out a word or drawing lines to create space. Since it also contains contextual analysis, the surrounding words, sentence structure, and grammar rules improve the recognition results. However, it is unsuitable for the project as it is primarily prepared for recognising natural languages.

### 2.4.2.3 GitHub Repositories

In GitHub, we can also find some open-source repositories that could be useful for extension in this project. GitHub user samkit-jain holds a repository of a tool they have built, a convolutional network project to recognise handwritten numbers and letters, written in Python and using the Tensorflow framework ${ }^{14}$. It is also unfit for this project because it only recognises individual characters.
math-handwriting-lib of user Glyphoid is a Java library developed for parsing and evaluating handwritten mathematical expressions ${ }^{15}$. It utilises stroke representations of handwritten symbols and supports different notation syntaxes for arithmetic, matrices, vectors, trigonometry, and others. Unfortunately, this project is outdated, and the choice of the Java language creates unnecessary complexity.

The tool most worthy of note found on GitHub is Seshat ${ }^{16}$, an open-source system for reconnising handwritten mathematical expressions that, given a sequence of strokes, converts them into expressions in $\mathrm{IATEX}_{\mathrm{E}}$ MathML and InkML [19] format. It is part of a PhD thesis by Francisco Álvaro, a former member of the PRHLT Research Centre at Universitat Politècnica de València. The system has participated in international competitions and has been awarded the best system

[^6]trained on the competition dataset in the 2014 International Conference on Frontiers in Handwriting Recognition (ICFHR). This guarantee of a good performance on the Competition on Recognition of On-line Handwritten Mathematical Expressions (CROHME) [53] use is what led to the decision to use this recogniser as a foundation, and connecting it to the system we developed.

## Chapter 3

## System Design

3.1 Desiderata ..... 20
3.2 Handwriting-only vs. Handwriting and Keyboard ..... 21
3.3 Interface Design ..... 21
3.4 Recognised symbols ..... 23
3.5 Modular Design ..... 24

This chapter focuses on the system design, defining the structure, organisation, conceptual requirements and design principles that ensure stability in the increase in complexity or scale. Section 3.1 defines the system's desired behaviour. The input mode choice is described in Section 3.2, p. 21. The principles guiding the interface design are illustrated in Section 3.3, p. 21. Section 3.4, p. 23 mentions the essential symbol support necessary for the project's success. The system's modular design and its advantages are explained in Section 3.5, p. 24. This chapter showcases the key features and functionalities that will enable the problem explained in Section 1.2, p. 2 to be solved and to ensure a successful implementation of the proposed solution explained in Chapter 4, p. 25.

### 3.1 Desiderata

Being an online handwriting recognition tool for mathematics, we did not want to develop a software tool that exclusively addressed the lack of tools considering the Calculational Method. Even though that is, in fact, the core of the problem we are addressing, this system should be helpful when dealing with other general mathematical annotations. The ultimate goal is for it to be possible to extend and integrate this tool into other kinds of mathematics-related applications that may eventually perceive mathematical structures like vectors and matrices, calculate equations or solve problems.

Ideally, the developed software tool should work for any operating system and be compatible with most peripherals. For now, the tool is developed in such a way to be used in the most common operating systems and be compatible with a tablet PC , as these computers provide the most natural
input method that mimics the experience of using pen and paper. Another reason why we prioritise the tablet PC is because of its portability, ability to write mathematics independently of location. This could be a crucial factor for a scientist that needs to annotate ideas suddenly.

Furthermore, in a perfect scenario, the user should be able to interact with the system by exclusively using their stylus on the tablet, executing all actions through touch and gestures, without needing to press physical buttons such as on a keyboard or a mouse. For this reason, we emphasised handwriting/drawing and touch. We developed a software tool that uses touch and drawing as the exclusive input method for the application, promoting interactivity and allowing users to naturally express their thoughts and ideas without the constraints of a keyboard.

### 3.2 Handwriting-only vs. Handwriting and Keyboard

Although making the tool focused on handwriting and touch-based interactions provides intuitiveness, accessibility, and makes the user more in tune with their thoughts, it does not come without compromises. Handwritten recognition often struggles with accuracy, may be slower than typing, is more challenging to edit and is even influenced by the quality of the tablet screen's pen detection. Even with this reasoning, because we want to move in a direction where the user is neither overwhelmed by the complex interface that a mixed approach would require nor by the learning curve that switching between both input methods involves, it has been decided to implement features only for handwriting. The model we are going for is different from a document model as we can find in Microsoft Word ${ }^{1}$ or Windows' built-in Notepad application, where the user is restricted to a grid-like environment, and instead, one focused on the drawing of the ink, not bound by strict formatting. This is further encouraged, as most features that the keyboard would fulfil can be implemented in a future project that handles gestures and gives them meaning in the sense that a gesture can translate to a set of button presses. Some of the buttons on the interface can be interacted with through shortcut keyboard binds, which can be important for users who run the tool on traditional PCs. Nevertheless, we focus on the tablet computer, so additional features should not require shortcut binds. For these reasons, we assume a handwriting-only approach for the implementation.

### 3.3 Interface Design

When planning the design for the client interface, to concede the user an enjoyable, valuable experience, some thought went into how the features are being presented. As most of the whiteboard's features were already implemented and presented understandably, only a little work went into changing the interface. However, some considerations were made to verify that the already existing and new functionalities satisfy all needs for the tool's goals.

[^7]

Figure 3.1: Demonstration of the implemented system's design and functionalities. A simple proof in the calculational style is demonstrated with the steps $a+b=c,=\{b=x\}$ and $a+x=c$

The whiteboard should be user-centred, which means the interface should be intuitive, allowing users to interact efficiently and understand the layout and the buttons logically and familiarly. Figure 3.1 shows how the current interface looks.

It is possible to argue that the whiteboard client interface is currently effectively user-centred because of the participant's behaviour in the user study described in Chapter 5, p. 43. We noticed that users usually skipped the instructions part where the required buttons for the study were explained, understanding each button's function by their iconography. While most buttons required for this project were already developed, we added a button that downloads the ${ }^{\mathrm{LAT}} \mathrm{E} X$ recognition that the user sees displayed in the top-right corner. Integrating this new button in the existing toolbar proved to be as intuitive as the other buttons, as there were no questions about how to download during the study by any of the users.

The user-centeredness is further complemented by the collaborative features of the whiteboard, meaning that two or more people may write simultaneously while sharing a single session. However, as this feature is irrelevant to the newly added recognition, it will not be discussed further.

The choice of placing the handwriting recognition on the top-right corner of the whiteboard seemed intuitive. In this case, we consider an adherence to the standard left-to-right writing pattern, which, allied to the space that the whiteboard already gave to the right of the toolbar, made for an appropriate place to display the recognition. Once the user starts writing, which will usually be on the left or in the middle of their device's screen, they will understand that the result will be displayed on the right, quickly confirming what they are writing and what is being recognised. The recognition is also shown in black to cause a contrast with the whiteboard's colour. This way, a user with any background can easily understand what is being shown.

Although the recognition tries to be processed as quickly as possible, there is some natural delay between the user's handwriting and the recognition results. In order to not confuse the user, we have added an indicative "Loading..." message in the same place where the recognition would be displayed whenever it is being processed. Adding this message prevents the user from thinking that the recognition is wrong rather than potentially being simply late.


Figure 3.2: Simplified example of a proof structure in the calculational style with A, B and C as steps and p and q as hints

Furthermore, it is vital to ensure that the application has a responsive design, meaning that it adapts to different devices' screen sizes. In this case, on smaller device screens, the toolbar is divided into two parts, leaving the right side free for the recognition to be shown.

### 3.3.1 Generation of the $\mathbf{N}$-Best Recognitions

Since the project's inception, the idea to provide the user with not merely a single recognition result, but rather a set of the most likely recognitions, was always kept in mind. We envisioned that this could be implemented in a drop-down selection menu that would allow users to quickly correct inaccurate or erroneous recognitions by clicking the option they found to be the most precise in a user-friendly format. The PRHLT Research Centre based in València has tackled this problem as of recently [50], with a paper that proposes a solution based on generating the N-best parse trees from 2D-PCFGs and generating hypergraphs [10] from these parse trees. However, as intriguing as this solution might have been, integrating these changes was deemed too complex, so it was ultimately concluded that the scope of this thesis would not allow for the implementation of this feature. Despite this decision, the concept remains a fascinating topic for future work that would provide a valuable asset to our own system and potentially contribute to the other recognition systems, paving the way for a better user experience.

### 3.4 Recognised symbols

While the developed system recognises and interprets various types of notations, it is crucial to clearly understand the expectations associated with the mathematical expressions that may be written. Clarifying this avoids potential misinterpretations and promotes results that contain valid meaning.

As shown in Section 2.1.4, p. 7, proofs in the calculational style follow a structure like the one presented in Figure 3.2. This means that besides recognising traditional mathematics like arithmetic, algebra, equations, inequalities, fractions, exponents and trigonometry, we also need to include support for propositional logic, as many proofs require the application of propositional
rules. All of these types of expressions need to be adequately recognised both in the statements (A, B and C in Figure 3.2, p. 23) and in the hints (p and q in Figure 3.2, p. 23).

### 3.5 Modular Design

Modular systems consist of self-contained units, each one with a specific function that should work independently and can be easily replaced. The apparent advantage of focusing on modularity is that, although it may involve a more complex development process, it allows for flexibility, scalability and ease of maintenance. In this case, we intended the system to be modular in different senses. It should be separated into different files that follow a hierarchy and functions that are well-documented. This approach ensures maintainability and scalability if users want to seamlessly extend the codebase or integrate the system with their existing applications. Breaking down the code into separate files, each serving a specific purpose, enhances the overall organisation and readability, allowing developers to modify specific components. Good documentation is undoubtedly valuable for developers since it details any constraints or assumptions associated with their usage.

Most importantly, it is modular in that Seshat is being used as the recogniser for the moment, but it is possible to replace it with another, new, better, more appropriate, or more recent recogniser of mathematics. In theory, all kinds of recognisers could be adapted to the current project, even those specialising in recognising natural language. As research progresses and new algorithms and models are developed, the community around proofs and mathematics can integrate their work into the existing framework, allowing the system to stay up-to-date with the latest technological innovations.

## Chapter 4

## Solution and Implementation

4.1 Architecture and Overview ..... 25
4.2 Whiteboard ..... 28
4.3 Seshat ..... 34
4.4 ExpressJS Server ..... 38

This chapter analyses the solution applied ${ }^{1}$, elucidating the implementation process. Beginning with an overview of the system's architecture in Section 4.1, we then dive into the functionality of the individual components and their communication so that a cohesive system is produced. Section 4.2, p. 28 details the implementation of the added features to the whiteboard. Section 4.3, p. 34 talks about the extension of the pre-existing recogniser Seshat. Finally, Section 4.4, p. 38 details the pivotal role of the server in the system.

### 4.1 Architecture and Overview

### 4.1.1 Client-Server Pattern

As illustrated by Figure 4.1, p. 26, the system is divided into three components that form an architecture in a Client-Server pattern. In this case, the whiteboard ${ }^{2}$ acts as the client that initiates requests to the server. An ExpressJS ${ }^{3}$ server waits for these, processes them, sends them to Seshat ${ }^{4}$ - the online handwriting recogniser serving as a sub-server or a service - and, after an exchange of messages between Seshat and the server, eventually returns an appropriate response to the client. Next, we enumerate and clarify the interactions of the system.

1. The whiteboard sends a POST request containing data with strokes to the server.
[^8]
## System Interaction Sequence Diagram



Figure 4.1: Sequence diagram detailing the system's architecture and the interactions between the three components
2. The server processes the request and its data and writes the strokes data to a file in an appropriate format.
3. The whiteboard sends a GET request to the server, requesting the recognition.
4. Upon receiving the request, the server calls the execution of Seshat and sends the stored file data.
5. Seshat processes the strokes data, recognises the mathematical expressions, and sends a list with the $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ results to the server.
6. The server sends the mathematical expressions to the whiteboard for displaying.

The next sections better explain the properties of each component and what types of messages are being exchanged.

### 4.1.2 Adapter Pattern

As mentioned in Section 3.5, p. 24, it is vital to design the system to be modular so that the recogniser can be substituted by a better recogniser once available. For this reason, the Adapter pattern was implemented. This design pattern creates a bridge between objects with incompatible interfaces to work in the same environment. So its purpose is to convert the interface of one class into another interface that is expected, promoting code reusability and loose coupling.

The Adapter pattern involves four components: the Client, the Target, the Adapter and the Adaptee. The Client attempts to perform actions which are impossible to do directly, as it is incompatible with the component it wants to interact with. As such, it must communicate with the Target interface. It represents the system's operations and is how we decouple the Client from the Adaptee. The Adapter allows the Adaptee to be effectively used by the Client by implementing the Target interface. The Adaptee is the component that performs the actual work, with methods that the Clients need to use.

In this case, the Client is the part of the system that interacts with the recognisers, the handler function for the GET requests. The Target is the AbstractRecogniser class that provides the interface that all recognisers must adhere to and implement. SeshatRecogniser is the Adapter for the Seshat online handwriting recogniser system, the Adaptee, to the AbstractRecogniser interface. Future recognisers will also be adapters, adjusting their systems to this interface. Figure 4.2, p. 28 represents these relationships for the current system.

Implementing the Adapter pattern provides flexibility and scalability. By defining a standard interface for recognisers, new ones can be easily added without changing the parts of the code that use them. The only change needed is creating an adapter for each new type of recogniser.


Figure 4.2: Adapter Diagram representing system's modular approach for accommodating multiple recognisers

### 4.2 Whiteboard

In order to understand what the whiteboard sends in its requests to the server, we first need to observe the internal structuring, i.e., how information in the whiteboard is kept, specifically, how to track what the user is writing and what actions they perform. As mentioned before, the whiteboard chosen is a NodeJS web application and contains the following file structure:


The "index.html" file defines the elements presented on the page - the crucial being HTML5's canvas element that serves as the pillar for the whiteboards' drawing capabilities - that are styled as specified in the "main.css" file. The Javascript files naturally present all the logic for interactivity and functionality.

The most important file is "whiteboard.js", which contains all the properties of the whiteboard object, the entity that effectively tracks what is happening on the canvas, and the added code for the new features. Next, we present an overview of the most important functions, where the bulk of the added logic resides.

```
OBJECT whiteboard:
    PROPERTIES:
    drawBuffer
```

```
    strokesArray
    timeoutId
loadWhiteboard():
    ON mouseup:
        IF tool IS ''pen', THEN
                clearTimeout(this.timeoutId)
                this.timeoutId = SET_TIMEOUT(
                    calculateStrokesArray AND refreshRecognition,
                    SESHAT_TIMEOUT)
        END IF
    other event listeners
calculateStrokesArray():
    Process drawBuffer into individual strokes
refreshRecognition():
    Generate an inkML string
    CALL getRecognition() to communicate with the server
    and display the contents
undo(), redo(), clear():
    CALL same actions as in mouseup event
```

The "drawBuffer" is an array of objects, each representing a drawing instruction, encapsulating all writing and erasing operations that transpired on the whiteboard up to a given moment. It includes properties such as the type of tool used like "pen" or "eraser", the tool's thickness represented by a number, the coordinates that were written, the colour, and a unique drawing identifier "drawId". In this case, these instructions only contain four coordinates representing where that part of the drawing has commenced, two intermediate coordinates, and where it has ended. The stroke, which is considered the ink trail drawn between consecutive "mousedown" and "mouseup" events, is identified by the "drawId". This means that a single stroke might be depicted by any number of elements in the "drawBuffer" array contingent on its length. However, every element shall contain the same "drawId", which is subsequently incremented when a new stroke starts. When the whiteboard is loaded, every "mouseup" event triggers two key functions:

1. The function "calculateStrokesArray" populates the "strokesArray" property, comprised of sub-arrays, each representing a stroke with all its pertinent coordinates. It distinguishes erasing actions from drawing actions and normalises all coordinates. This normalisation step is necessary, as the whiteboard may contain negative coordinates should the user shift the canvas towards the negative quadrants of the imaginary $\mathrm{X}-\mathrm{Y}$ axis with the hand tool.
2. The "refreshRecognition" function is responsible for updating the HTML element containing the recognition. To do that, first it calls the "getRecognition" function that initiates the message exchange with the server.

These steps described are also taken whenever there is the action to undo or redo a user's stroke, clearing the whiteboard, or erasing part of the stroke. These functions are inserted in the context of a timeout, meaning that they are only called if the user has not done any actions to reset the timeout, like continuing to draw or performing any actions that change the ink on the screen. This timeout is set for 1100 ms , so users instantly get the recognition result while conceding the system a considerable performance boost.

### 4.2.1 InkML

In order to understand the interaction between the whiteboard and the server upon calling the "getRecognition" function, it is crucial to understand what InkML is.

InkML is a standard XML-based markup language developed by the W3C ${ }^{5}$ used to represent digital ink data that bridges the physical act of writing and its digital representation. It expresses information on the paths the pen takes, the pressure, tilt, and potentially the time at each point. Metadata and contextual information can also be attached to the ink, such as the author, the writing instrument, the thickness of the line, the device and environment in which the ink was captured, and other relevant information.

Listing 4.1 shows an example InkML file that stores information about the ink from a user that wrote an " X " on the screen. The example is simplified, containing only the coordinates of the two distinct traces that construct the " $X$ " symbol. A space separates each pair of coordinates, and each trace is identified to be unique.

Listing 4.1: InkML example forming " X "

```
<?xml version='`1.0', encoding='`UTF-8''?>
<ink xmlns=''http://www.w3.org/2003/InkML',>
    <trace xml:id="st 1">10.0,10.0 20.0,20.0 30.0,30.0}40.0,40.0 50.0,50.0</
        trace>
    <trace xml:id="st2">10.0,50.0 20.0,40.0 30.0,30.0
        trace>
</ink>
```


### 4.2.1.1 MathML

The excerpt in Listing 4.2 shows an actual, complete illustration of the structure and content an InkML file contains when output by Seshat. In this instance, the expression $a+b$ serves as an example. The annotations in the beginning indicate the ground truth in a tree-like structure, and the traceGroups at the file's end contain the symbol and the strokes that belong to it. The next aspect that requires elucidation is how MathML functions.

Listing 4.2: Generated InkML file example of "a+b"

```
<ink xmlns="http://www.w3.org/2003/InkML">
<annotation type="UI"></annotation >
    5https://www.w3.org/TR/InkML/
```

```
<annotationXML type="truth" encoding="Content-MathML">
<math xmlns='http ://www.w3.org/1998/Math/MathML'>
<mrow>
<mi xml:id="a_6">a</mi>
<mrow>
<mo xml:id="+_7">+</mo>
<mi xml:id="b_8">b</mi>
</mrow>
</mrow>
</math>
</annotationXML>
<trace id="0">
</trace>
<trace id="1">
...
</trace>
<trace id="2">
...
</trace>
.
<annotation type="truth ">Segmentation </ annotation >
<traceGroup xml:id="6">
    <annotation type="truth ">a</ annotation>
    <traceView traceDataRef="0"/>
    <annotationXML href="a_6"/></traceGroup >
<traceGroup xml:id="7">
    <annotation type="truth ">+</ annotation >
    <traceView traceDataRef="1"/>
    <traceView traceDataRef="2"/>
    <annotationXML href="+_7"/></traceGroup >
</ink>
```

MathML ${ }^{6}$, a markup language developed by W3C, uses an XML-based syntax to encapsulate mathematical notations in their structure and content. The structure starts with a "math" tag followed by a "mrow" tag and its child elements. In this case, "mrow' represents groups of sub-expressions. However, Seshat, unlike other MathML documents, considers the smallest sub-expression possible, assigning to every "mrow" two children entities, making it a binary tree, where a minimum of one child will invariably be another "mrow" element, except for the deepest elements in the tree. In the shown example, "mi" elements correspond to variables and "mo" elements represent operators. MathML tag support extends to fractions, exponents, and other elements essential for formulating traditional mathematical expressions. Figure 4.3, p. 32 translates the Listing 4.2, p. 30 into a visual binary tree.

[^9]

Figure 4.3: Diagram containing the respective binary trees structure diagram of the MathML equivalent of the expression $a+b$

### 4.2.2 Display

Only two libraries were considered regarding the display of the $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ equivalent of the handwriting received by the whiteboard after the requests are sent. MathJax ${ }^{7}$ is a JavaScript library that allows for displaying mathematical expressions in browsers, supporting typesetting with LETEX and MathML. It is used partly in academic and scientific spaces due to broad browser compatibility. However, it falls short compared to $\mathrm{KaTeX}^{8}$ - an open-source JavaScript library with a similar purpose - because it is generally faster and more straightforward, rendering math synchronously. MathJax does support a broader range of $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ commands, but such support is not required.

Therefore, a "div" element was created and styled for the recognition result's display, and it changes dynamically on every recognition update rendered by Katex.

### 4.2.3 Erase Feature

Unlike the undo, redo and clear features, where integration with the added code was relatively straightforward, adapting the eraser involved some attention. The challenge is rooted in its distinct interaction with the "drawBuffer" structure. As mentioned, this array registers the erasures by overlapping white ink on top of prior strokes, effectively adding new elements instead of deleting or modifying the existing instructions.

The refinement of the strokes array to disregard erased ink happens inside the "generateStrokesArray" function. This function receives a map of drawn strokes in conjunction with a set of eraser points, each with a thickness. Each ink point within erasing range of any eraser point is identified and removed from the stroke. Empty strokes at the end of the process are also eliminated. The procedure to ascertain whether an ink point falls within the eraser's range is done with a function that computes the Euclidian distance between two points.

[^10]

Figure 4.4: Example of the approach to recognise free text by enclosing it in a box

### 4.2.4 Download $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ button

Lastly, a button was added to the toolbar that allows users to export their handwritten notations into a downloadable $\mathrm{ET}_{\mathrm{E}} \mathrm{X}$ format. For instance, if the user writes the expression, $a \wedge b$, on the display, they would see exactly that on the recognition results, as it would be rendered to $\mathrm{EAT}_{\mathrm{E}} \mathrm{XI}$ In contrast, the internal representation would be "a lwedge b" instead. This converted LATEX format is provided to the users as a .tex file when they press the newly added button.

### 4.2.5 Limitations and Assumptions

It is important to note that the system does not make any assumptions about what is being written. This means that the recogniser does take into account the context in which the mathematical exists and therefore makes better predictions. However, it does not interpret the user's writing semantically, and it does not have the objective of validating the written proofs in any way. It only aims to make the Calculational Method more accessible, converting handwriting into a more convenient format. In order to achieve semantic validation, a work would need to exist that first structured the written ink into an expression.

Currently, the system is focused on supporting proofs containing mathematical expressions in the statements and hints. Although the tool is technically prepared to receive free text, as the recogniser is specialised for mathematics, it was not adapted to distinguish natural language from mathematical expressions, and, as seen in Chapter 2, p. 5, no current recogniser can perform both detections with excellency.

As such, two solutions for the interface were considered. The first would involve creating a button that could be toggled on and off. Ink written with the button toggled would represent free text to be ignored, while ink with the button off would represent ink to be sent to the recogniser. However, this solution is prone to user error, where the user may potentially forget to interact with the button, for example, causing frustration, and considering the lack of automation that this approach carries, the solution was also thought to disrupt the workflow of the users, discouraging the synchrony of the user's cognitive state with the content of the handwriting. A second option could reside in identifying free text with the handwriting itself, for example, by enclosing it in a square as seen in Figure 4.4, but this would involve retraining Seshat's RNN, which is impossible for reasons that will be explained in Section 4.3, p. 34. None of the solutions were implemented, leaving the second as an encouragement for future work.

### 4.3 Seshat

The scope of this thesis would need to be larger to consider making a handwritten mathematics recogniser from scratch. Instead, we contribute to the online handwriting community in the sense that there will be a complement to that which is already available.

The recogniser - the backend of this project - is a library that provides easy access for other applications to connect to this module and use its features independently of how the results are used and displayed to the end user. Seshat, the chosen online handwriting recogniser, whose name is inspired by the Goddess of Writing in Egyptian mythology, is written in C++ and integrates and adapts the open-source library $R N N L i b^{9}$, and the Xerces- $C^{10}$ library that parses InkML.

RNNLib is a recurrent neural network library applicable to various types of spatiotemporal data, particularly useful in the area of handwriting recognition, that implements Bidirectional Long Short-Term Memory, which makes it an excellent tool for labelling and classifying data with spatiotemporal dimensions. Xerces-C is an XML parser that offers ways for parsing, generating, manipulating, and validating XML documents.

To use Seshat, one needs to compile it and pass some parameters, including the configuration file, the InkML file containing the math expression, and other optional flags for the output file, saving an image representing the input expression and the tree representing the expression. For example, one could run the command

```
./seshat \
-c Config/CONFIG \
-i SampleMathExps/exp.inkml \
-o out.inkml \
-r render.pgm \
-d out.dot
```

that takes the file called "exp.inkml" as input.
The motivation for its conception is explained in a paper that discusses the challenges of automatic recognition of mathematical expressions, which arises from the ambiguities at different levels [2]. They present an integrated approach combining different stochastic sources to determine the most likely mathematical expression globally, allowing for simultaneous optimization of symbol segmentation, recognition, and structural analysis.

The open-source code follows the following structure:

```
/seshat
    _ seshat.cc
    [_cellcyk.cc
    ._tablecyk.cc
    . _stroke.cc
```

[^11]

The "Config" folder contains various configurations. In the "Grammar" folder, the file "mathexp.gram" defines a probability context-free grammar. It assigns weights to a set of mathematical symbols, defined in symbols.types, and syntax structures that model the possible constructions. The file helps in understanding and generating structured mathematical expressions. These grammar rules and their associated probabilities are used to generate expressions based on their likelihood of occurrence. The "SymRec" folder contains configurations of the recognition of individual symbols. "rnnlib4seshat" is the directory that contains the adaptation of the "RNNLib" library.

Regarding the codebase, some of the most important files are shown in the tree. "seshat.cc" is the main file that ties all the components together. The "cellcyk.cc" and "tablecyk.cc" files are related to the CYK (Cocke-Younger-Kasami) [18] algorithm that determines whether a string can be derived from a context-free grammar. The file "stroke.cc" defines a stroke as a segment or series of segments made by a writing instrument without lifting it from the surface. The "segmentation.cc" file defines a class that implements a segmentation model based on Gaussian Mixture Models and provides functions to load the model, calculate statistical values for segmentation hypothesis and compute the probability of the hypothesis being correct. Lastly, the "meparser.cc" file performs the recognition and parsing of the expression using the provided grammar.

It is important to note that Seshat interprets the input file's contents as mathematical expressions in a single line. So if data from multiple expressions were passed to Seshat, its processing would produce a recognition far from expected. As the Calculational Method implies writing multiple lines, one for each step of the proof, we had to design a way to segment each line and recognise them individually. This will be further discussed in subsection 4.4.1.

The annotations in the MathML ground truth extracted from the InkML file that Seshat outputs underwent some modifications and corrections. Every element was mistakenly identified as a variable and assigned the tag "mi". However, these elements were, in fact, operators or number literals. Then, to accommodate the various lines that a proof of Calculational Method requires, the InkML files were adapted to incorporate multiple "math" tags, each with corresponding annotations. Presently, these tags symbolise individual lines within the proof and carry a property "data-type" indicating whether that line is a statement or a hint.

### 4.3.1 Supported Symbols and Limitations

An overview of the structure of the "mathexp.gram" file should precede an explanation of what symbols are recognised and how the new symbols were added.

Listing 4.3: Example mathexp.gram file

```
2dot
ArrTerm
Arrow
Bar
START
Term
Sym
UnTerm
TermSub
TermSup
TermSSE
PTERM
0.05751515 Digit 0 0
0.18358371 Digit 1 1
0.34127313 Digit 2 2
0.12341273 Digit 3 3
0.10621062 LetterCap X X
0.03330333 LetterCap Y Y
0.18756269 Greek \alpha \alpha
0.13640922 Greek \beta \beta
0.00990375 OpBin \div \div
0.03898731 OpBin \times \times
0.23678337 OpBin equal =
PBIN
0.06462269 H Term Term OpTerm '`$1 $2', M
0.03947800 H Term TermSup OpTerm '`$1 $2', M
```

As we can see in Listing 4.3, the file commences with an enumeration of the nonterminals followed by the first production rule. In this case, it may take the form of a generic term, a more specific term or a symbol. This distinction between generic and specific terms comes from the possible derivations that each possesses. Specific terms imply that their derivations have a unique spacial relationship, such as subscript or superscript. Following the list of nonterminals is the list of probabilities each nonterminal has of being derived to the following terminal. Their corresponding representation in $\mathrm{EA}_{\mathrm{E}} \mathrm{X}$ accompanies this. Finally, the file contains a list of probabilistic production rules. Let us clarify further with the following example:

### 0.06462269 H Term Term OpTerm "\$1 \$2" M

The probability of the rule happening is succeeded by the type of relation the derivation has from the first nonterminal. In this particular instance, "H" denotes a horizontal relationship. Following this, we see that "Term" can be derived into another "Term" followed by an "OpTerm". Specifically, "OpTerm" implies the presence of a new term which can be further derived into a binary operator and another term. In quotations, we see how these nonterminals are output, with " $\$ 1$ " substituted by "Term" and " $\$ 2$ " substituted by "OpTerm", separated by a space. The final character, "M", delineates the direction of writing, meaning middle in this context. This character may be " B " or " A " for subscript and superscript relations.

The newly added symbols can be seen in the listing 4.4.
Listing 4.4: Extension of mathexp.gram file

```
0.50000000 OpBin v \vee
0.50000000 OpBin V \vee
0.90000000 OpBin 1 \wedge
1.00000000 OpUn 7 \neg
1.00000000 OpBin u \cup
1.00000000 Sym equal =
0.70000000 Sym \1t \1t
0.80000000 Sym \gt \gt
0.90000000 Sym \leq \leq
0.80000000 Sym \geq \geq
```

Seshat already supports many types of symbols and expressions, including numbers, letters, greek characters, symbols for trigonometry, set theory, integrals, sums, equalities and inequalities. However, analysing the most common symbols in the Calculational Method, we concluded that it lacks some notation that is important to include and is supported by $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ some of which are presented in the following list:

$$
\vee, \wedge, \equiv, \cong, \Leftarrow, \Rightarrow, \uparrow, \downarrow, \square, \cap, \cup, \Pi
$$

The methodology we employed to incorporate additional symbols is closely linked to the impossibility of retraining Seshat's recognition model. This conclusions originated from a contact with Seshat's authoor himself. Although the code that manages the input and output of expressions is open-source, generating the weights for the probabilistic context-free grammar proves challenging. Although Alvaro et al. explain how the weights were calculated in their paper, the actual code to reproduce these values has yet to be made public [2].

Therefore, Seshat's model to train the RNN and apply the weights to the probabilistic contextfree grammar remains proprietary. As such, it is impossible to introduce new symbols directly by extending the grammar. Instead, we resorted to giving already existing symbols a probability of being recognised as different symbols. This causes complications, as the recognition's effectiveness suffers from a lack of the surrounding context. For example, it was possible to add $\vee$ whenever a " v " is detected and $\mathrm{a} \neg$ in the case of " 7 ", but predictably, the values are assigned
based on empirical evidence rather than sound reasoning or training. Furthermore, as the hints do not follow the syntax of traditional mathematics, the $=$ symbol and the equivalent for inequalities needed to be added as standalone symbols.

In an optimal scenario, users should also be able to add new operators spontaneously to be recognised. They could assign these operators an arbitrary meaning by interacting with the interface. For instance, users might define a ternary $\nabla$ operator that takes precedence over the multiplication operator. Nevertheless, something like this would require retraining Seshat's model, which appears impractical in the current circumstances.

### 4.4 ExpressJS Server

The connecting element between the interface and the recogniser is a server built with ExpressJS, a backend framework for web applications and APIs in Node.js, which provides features for routing, session management, cookies and form data parsing and general handling of HTTP requests. This technology has been chosen for the project as it is known for its simplicity, flexibility in terms of architecture and design, a large ecosystem with packages for most needs and easy integration with any frontend. As explained previously, ExpressJS is tasked with serving the requests from the whiteboard, relaying them to the assigned recogniser and processing data.

The code follows this structure:

```
/Express-Server
_ index.js
_ /adapters
        _ _seshatRecogniser.js
- /handlers
    _handleGet.js
    _ handlePost.js
    /routes
    _ seshat.js
    /utils
    _ constants.js
    _ fileUtils.js
    _ _traceUtils.js
        _xmlUtils.js
. .
```

The "index.js" is the application's entry point that starts the server. The "seshatRecogniser.js" file and the "adapters" folder are the implementations of the already explained Adapter pattern. The main functions in the "handlers" folder handle GET and POST requests received on the "/seshat" endpoint, defined in the "seshat.js" file. The "utils" directory contains the constant values used throughout the tool and functions related to file operations, XML parsing and manipulation, and general calculations.

Among the steps in processing the data that the server receives, line segmentation stands out as one of the most complex and is crucial to understand how the requests are handled.

### 4.4.1 Line Segmentation

In this context, line segmentation refers to the partition of several handwritten expressions into individual lines. The precision of this segmentation impacts the recognition significantly, as separate lines being treated as one continuous expression may result in inaccurate interpretations. The challenge of this task lies in different factors. Among them are the inconsistent spacing between lines depending on the writer's style, lines of text that may overlap, tilted handwriting, and noise or smudges that can be misinterpreted as a part of a line.

To implement this segmentation, several methods were considered, which will be described next.

### 4.4.1.1 RNN-based multi-line segmentation using special Dynamic Threshold Distance

In their paper, Yakovchuk et al. [66] elucidate a solution that utilizes a BLSTM neural network with a Connectionist Temporal Classification (CTC) [29] output layer for character recognition. To overcome ambiguity in some scenarios, the authors proposed a Dynamic Threshold Distance (DTD) that calculates the preferred line spacing based on empirical data and user interface preferences. The algorithm used proved to be successful as it segmented correctly roughly $93 \%$ of the expressions on the 2016 CROHME competition. The authors also put forth the pseudocode for their unique line segmentation algorithm, incorporating the DTD. However, they need to be more detailed in describing how some algorithm functions work, and define some aspects of the DTD's computation more clearly, such as the ambiguously presented 'fontSize' variable. Ultimately, this lack of clarity drove the decision to set aside this method for the project.

### 4.4.1.2 Linear Interpolation based on Pen Thickness

A quick, informal experiment was done to investigate a potential correlation between a chosen pen thickness and the appropriate vertical spacing required for legibility in written symbols. The idea is that characters that require curving to be drawn - like the characters "a", "u" or "o" - might need a larger vertical space for accurate representation when written with greater thickness. As the pen's thickness increases, the curves of characters occupy more vertical space, which affects legibility if not adjusted accordingly, appearing distorted. Figure 4.5, p. 40 represents what is meant visually.

As such, an interpolation was attempted to represent the relationship between the thickness and the vertical distance. In the presented interface, the thickness slider represents internal values ranging from 1 to 50 , with no specific units. These values were compared with the minimum vertical spacing that each line would require to be registered as another expression, which is essentially expressed as the difference in their coordinates, also as internal data with no units. However, it is essential to note that this data is purely a preliminary approximation employed as an estimate for testing purposes. Had this segmentation method yielded superior results, a user study would have

## $a \longrightarrow a$

Figure 4.5: Character "a" written with pens of different thicknesses, demonstrating a lower legibility of curved strokes with the increase in thickness
been conducted to represent users' spacing preferences authentically. Unfortunately, this was not the case; thus, this approach was not incorporated into the project.

Table 4.1 shows the obtained data, whereas Equation 4.1 lays out the interpolation equation derived, and Figure 4.6, p. 41 shows that interpolation graphically.

$$
\begin{equation*}
y=8.42577168054875 * x-12.881920627143524 \tag{4.1}
\end{equation*}
$$

### 4.4.1.3 Fixed Line Spacing

The current approach relies on presuming a constant distance between each line and entrusting the user with the responsibility of leaving enough space between them. In essence, the distance is determined by averaging the Y-coordinate of all points comprising each stroke of every symbol, creating a midpoint for each of them, respectively. After that, based on a predetermined threshold, every symbol whose midpoint is vertically farther than the threshold is deemed to constitute a new line and represented as such in the interface. Figure 4.7, p. 41 represents these midpoints and their distance.

Naturally, this solution is imperfect, as the user may write with a thicker pen, write a taller expression or use fractions and exponents further away than expected, making the recognition inaccurate. This solution was adopted, albeit encouraged as an improvement in future work.

| Pen Thickness | Vertical Spacing |
| ---: | ---: |
| 1 | 5 |
| 10 | 60 |
| 20 | 130 |
| 30 | 280 |
| 40 | 320 |
| 50 | 400 |

Table 4.1: Table showing the noted values of the informal experiment that compare a potential relation between pen thickness and the minimum needed vertical spacing


Figure 4.6: Diagram representing the linear interpolation between thickness and vertical spacing of the values in table 4.1


Figure 4.7: Simplified representation of the distance calculation based on the average Y value used in line segmentation

### 4.4.2 GET and POST handlers

Finally, we explain how the server handles incoming requests and detail its internal processing. The POST request handler receives a string in InkML format representing the current strokes drawn on the whiteboard. This request acts as the catalyst for the series of message exchanges that follow. The server then parses the data by employing the "xml2js" library, which allows for the conversion of XML data into JavaScript objects. After that, the coordinates are extracted from every trace and their average $Y$ coordinate is calculated, as shown in Section 4.4.1.3, p. 40. Following the extraction, the data is written to local temporary files. To do this, the "createInkMLObjects" function analyses all trace information, groups them based on which line they belong to, formats the X-Y coordinate pairs and wraps them in XML elements. Finally, it generates an array of strings, wrapping all elements in their respective InkML formats, which is finally returned. Temporary files store these strings, each dedicated to a single line. The most recent files are relayed to the whiteboard, which signals the next steps.

As explained by the interaction seen in Figure 4.1, p. 26, what follows is a GET request that activates the recognition process. In this case, the "seshatRecogniser" class starts by assembling shell commands for executing Seshat. These commands specify parameters such as the configuration files and the output format, all of which adopt default values. After executing these commands, the temporary files are converted into recognised files, each representing a line, with strokes assigned to their respective symbols, also in InkML format.

The second stage of this handler involves merging all of the resultant output InkML files into a single one. Initially, the most recently written files are parsed, and the information of all traces is kept internally. After that, the "modifyIDsToUnique" function alters the object with the trace information, ensuring that every trace possesses a unique ID and correctly assigning them to their corresponding trace groupings.

With this object, the final stage is paved, where a new file is created. It encapsulates every line inside "math" tags and the trace information along with the appropriate annotations, as seen in Listing 4.2, p. 30. Creating a comprehensive InkML file is critical as it may be used as input for future research on the structuring and manipulation of handwriting ink. However, it is vital to be aware that integrating multiple "math" tags into the InkML file breaks its XML specification. Despite this, it was concluded that this method is preferable, as it preserves the remaining InkML's specification while allowing for multiple lines to be present and having each line contain the "mrow" tag tree structure that Seshat uses, even though, as explained previously, "mrow" tags are typically the element that separates various expressions. The final result is sent to the whiteboard as a response used in the display.

## Chapter 5

## User Study and Validation

5.1 Validation Goals ..... 43
5.2 User Study Instructions and Tasks ..... 44
5.3 Post-Experiment Questionnaire ..... 44
5.4 Methodology and Rationale ..... 46
5.5 Experimental Parameters ..... 48
5.6 Results ..... 48
5.7 Threats to Validity ..... 60

This chapter presents the performed user study aimed at validating the tool shown and assessing user satisfaction and usability. In Section 5.1, the goals attempting to be achieved are explained. Section 5.2, p. 44 provides the instructions presented to the participants to guide them during the experiment. Section 5.3, p. 44 shows the questionnaire presented to the participants that collect their feedback on the system's performance. The methodological approach and rationale of the study are elaborated in Section 5.4, p. 46. The experiments' conditions are further detailed in Section 5.5, p. 48. Section 5.6, p. 48 outlines all results, interpretations and conclusions from the study. Finally, Section 5.7 , p. 60 addresses potential threats to the study's findings.

### 5.1 Validation Goals

In conducting the user study, our primary objective was to appraise the efficacy of the developed tool, focusing on validating its usefulness, accuracy and efficiency for writing proofs in the calculational style. We also aim to determine the intuitiveness of our system's design and whether it provides a satisfying user experience. Since all users should have different handwriting styles, each one with its particularities, successful results should indicate efficiency in the recognition. This data could also be a valuable addition to insert in the learning process of the recogniser.

### 5.2 User Study Instructions and Tasks

In this user study, the participants were instructed to read the user study guide carefully (see Appendix A) and follow it exactly, which comprises three distinct steps:

1. The participants were asked to go through a tutorial to familiarise themselves with the tool. This initial exercise involved writing the expression $a+b=c$ on the whiteboard and modifying that expression to be $a+x=y$. To help them complete this preliminary task, some instructions were given to them that serve as a guide through the interface and its functionalities. In particular, they were introduced to the essential operations of drawing, adjusting the whiteboard's view, and erasing or removing the strokes. Users were also informed to monitor their input's real-time recognition on the interface's top-right corner, as explained in section 3.3. Once the users confirmed that they understood the functionalities and how to interact with the tool, they were asked to move on to the proper tasks.
2. In this part, the guide instructs the participants to transcribe two proofs in the calculational style onto the tool.

The first proof estimates a rough difference between 256 and 367 , concluding that $367-$ $256<200$. They were asked to give enough spacing between each line and advised to ignore the free text, leaving it blank, as it is not being recognised by the tool, which interprets every stroke as part of a mathematical expression. This proof consists of four short statements, the last being the proof's conclusion and three hints. The last hint that justifies that $400-$ $200=200$ contains the text "arithmetic" inside the brackets. This is an example of cases where the participants should write empty brackets instead. The complete proof can be seen in Figure 5.1, p. 45.

The second proof comprises a series of logical equivalences and simplifications commonly used in propositional logic. It is comprised of five statements and four hints, all of them containing exclusively free text. Recognising that the proof may be somewhat hard to follow during writing, participants were advised to copy the proof up to and including the third statement. The complete proof can be seen in Figure 5.2, p. 45. However, the participants were only asked to write that which is above the black horizontal line.
3. The participants were instructed to fill out a form before starting the warm-up and after completing each task.

### 5.3 Post-Experiment Questionnaire

The survey presented to the users (see Appendix B) intends to gather demographic data and obtain the participants' opinions on the developed tool. Before filling out the survey, the participants must acknowledge and agree that all the data collected in the study will be processed, stored and

| 367-256 |  |  |
| :---: | :---: | :---: |
| < | \{ | $367<400$ |
| 400-256 |  |  |
| $<$ | \{ | $200<256$ |
| 400-200 |  |  |
| $=$ | \{ | arithmetic |
| 200 |  |  |

Figure 5.1: Proof shown in the first task of the user study

```
        \((p \wedge q) \vee(p \wedge r)\)
    \(=\{\quad\{\quad\) distributivity of disjunction over conjunction (7.13) \(\}\)
        \(((p \wedge q) \vee p) \wedge((p \wedge q) \vee r)\)
    \(=\quad\{\quad\) absorption (see above) \(\}\)
        \(p \wedge((p \wedge q) \vee r)\)
            \(=\{\quad\{\quad\) distributivity of disjunction over conjunction
            (symmetric version) \}
        \(p \wedge(p \vee r) \wedge(q \vee r)\)
    \(=\{\) absorption \(\}\)
        \(p \wedge(q \vee r)\).
```

Figure 5.2: Proof shown in the second task of the user study. The black line depicts where users should stop writing the proof. In other words, what is written by the participants is found above the black line.
used for academic and scientific purposes. Participation is entirely voluntary, anonymous, and the participant is free to withdraw at any time without penalty, in which case they are expected to terminate their involvement in the study immediately.

When gathering demographic data and technical details, users were asked to indicate their age, gender, the specific hardware and peripherals employed during the experiment, and the operating system in use. Furthermore, they were required to rate their digital literacy level, familiarity with other handwriting recognition tools, and their most frequently used browser.

Moving to the experimental component, participants were prompted to evaluate for both proofs how easy was the writing process, how accurate the recognition was, the extent of rewriting necessary, and compare the complexity of manually typing the $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ equivalent of the proof versus using the proposed system. Additionally, we encouraged participants to compare the ease of handwriting the proof on a traditional PC, with a mouse, versus a tablet PC, or vice-versa, depending on the device they used.

The concluding part of the form poses questions about usability, particularly commonly asked questions from the System Usability Scale [14]. They were also offered the opportunity to provide observations and improvement suggestions.

### 5.4 Methodology and Rationale

In this section, we explain the design of the user study and the reasons behind the decisions made in the formulation of the questions.

The user study guide introduces the participants to the study's purpose and explains what is anticipated of their involvement. Since we want to evaluate the tool rather than the participants' understanding or interpretation of mathematical proofs presented in the calculational style, we have a basic comprehension of mathematics and propositional logic as prerequisites for participation.

To start, we ask the volunteers to fill out the first section of the provided form. Filling this section before moving forward to the other tasks is essential, as the participants should understand the implications of their participation. After collecting primary demographic data, users must fill in information about their digital literacy and knowledge of other handwriting recognition tools before interacting with our tool. Digital literacy is vital to ascertain, as users with a lower digital literacy might struggle to understand the tool's capabilities, features and outputs. Additionally, it is important also to gauge the experience users have with other handwriting tools, as it may help identify experience or familiarity biases [49, 13].

These biases refer to the influence on the users' perspective or reaction to something new. In this case, they may expect the tool to function in a certain way, overestimating or underestimating the tool comparatively. Participants may demonstrate a preference for a tool they already know, and not necessarily because it is better or more efficient than the proposed tool.

The following questions related to the environment also posed before the actual tasks, are essential to obtain at this stage. In this case, it is anticipated that users using a tablet PC equipped
with a digital pen have a better overall experience with the tool. This can be attributed to superior suitability for handwriting compared to users who use the tool in a tablet with no digital pen. Users using a traditional PC instead, with a mouse or trackpad, may expect more inconsistent recognition results due to poor handwriting quality. Obtaining information about the operating system in use and the most regularly used browser helps us guarantee that the tool is compatible with the most prevalent settings.

Once the participant has responded to these preliminary questions, they should complete the first task and address the associated questions. After completion, they should move to the second and complete the respective form questions. We divide the process in this manner instead of having the user answer the questions for both proofs so that the user can immediately record their experience. This approach helps tackle the recency effect [46] and overlap between the two tasks, ensuring that the feedback is independent and remains uncompromised.

The concluding segment focusing on usability is placed at the end of the interaction. This positioning allows users to form an opinion after having interacted enough with the system, evaluating the system as a whole, and potentially providing observations. It is relevant to mention that we integrate an attention check [35] midway.

Attention checks are questions embedded in the survey that require more than a glance to answer correctly. They aim to ensure that the participant is actively engaged and paying attention to the instructions. In this instance, it takes the form of a nonsensical question - "For this option, please select "Strongly Agree"'. The whole form entry is invalid if the user presses any other answer. Implementing such checks helps ensure high data quality and screen out inattentive participants.

### 5.4.1 Likert Scale

In order to understand the rationale behind the formulation of the questions, it is important to explain the Likert scale [40], a helpful model to capture people's opinions and perceptions, using a series of statements that measure the degree to which people agree with a particular topic. The 5-point scale used ranges from extreme disagreement to extreme agreement and is commonly employed in various types of surveys and feedback forms, making it useful in different scientific contexts. The questions presented in the Likert scale should maintain a neutral stance, meaning that they should not indicate the preferred response by the form's creator, if applicable. For this reason, most questions that seek the participants' opinions range from "strongly agree" to "strongly disagree". This even applies to the question about comparing the complexity of manually typing each proof and using the proposed tool. Although it is not presented in a numeric, linear scale, has still offers five response options. Standardising responses to the same scale helps the data analysis process, especially when examining the correlation between responses shown in Section 5.6.4, p. 58.

### 5.5 Experimental Parameters

The user study was performed with the following parameters:

- Participants: 9 of the 12 participants that were part of the experiment are all enrolled in higher education, with 6 being students in the fields of Computer Science and Informatics Engineering at Universidade do Porto. 11 out of the 12 participants fall in the 18-24 age bracket, while one participant is in the age range of $25-34$. In terms of gender distribution, $25 \%$ of the sample identifies as female, whereas the rest considers themselves male. The participants all confirmed that they have a foundational understanding of traditional mathematics and proofs, particularly inequalities, the critical element of proof 1. However, some participants shared that their comprehension of propositional logic, relevant in proof 2, could be stronger comparatively. The rationale behind the selection process for these participants was based on choosing volunteers from Porto who displayed potential interest in pursuing careers as scientists, researchers or educators in the future and that are at least moderately fluent in English. We ensured that every participant provided verbal consent before formally giving their consent through the form.
- Environment: All experiments were conducted in person with a personal computer showing the participants the guide and the form. The participants were given a Samsung Galaxy Note 2014 Edition tablet with a pen to interact with the whiteboard. The participants accessed the tool remotely with a desktop PC acting as the server. Performing the user study face-to-face allowed for close observation and enabled straightforward clarification of any questions raised by the participants.
- Duration: The participants were given ample time to complete the user study, averaging a completion time of approximately 20 minutes per participant. The completion time of each proof was recorded, starting from the moment the user began pressing the pen onto the device's screen. Further information can be found in Section 5.6.2, p. 56.


### 5.6 Results

In this section, we analyse the results obtained from conducting the described user study along with their interpretations.

In the questionnaire section related to demography and environment, one of the questions sought the users' self-assessment of their digital literacy. As shown by Figure 5.3, p. 49, most users rate their proficiency as mostly "strong" or "very strong", which suggests a reliable basis for the subsequent responses regarding the proposed system.

However, regarding the acquaintance level with other handwriting recognition tools of mathematical expressions, most participants reported complete or slight familiarity, which can be seen on Figure 5.4, p. 49.


Figure 5.3: Bar plot of the participants' ratings relative to their self-assessed digital literacy


Figure 5.4: Bar plot of the participants' familiarity level with other handwriting recognition tools of mathematical expressions


Figure 5.5: Pie chart representing the participants' most regularly used browsers

Figure 5.5 visually represents the browsers participants use the most regularly. As the data indicates, over half of the participants use Google Chrome, while the remaining users are split between the Brave and Mozilla Firefox browsers. Since the proposed tool was developed and tested in both of these environments, and the performed experience was carried out in Google Chrome, we can presume that the tool is supported in the most popular platforms and, therefore, can be experienced by a broad set of users.

### 5.6.1 Tasks

Regarding the feedback participants provided related to the proofs they have written, we intend to analyse both for every question (see Figure 5.1, p. 45 and Figure 5.2, p. 45).

When the participants were solicited to rate the difficulty level of writing proof 1 , most users reported it to be "extremely easy" or "considerably easy", as seen in Figure 5.6, p. 51. This allows us to deduce that any potential frustrations, errors or inaccuracies in the recognition process are likely not linked to the complexity of transcribing the proof itself.

When comparing the difficulty of writing proof 2 and proof 1, which can be seen in Figure 5.7 , p. 51, we can see that the results are not so straightforward. Despite a substantial majority considering the proof to be between "extremely easy" and "considerably easy", it should not be overlooked that two participants perceived it as "considerably hard".

After transcribing the proofs, the participants predominantly deemed the tool to be "neither accurate nor inaccurate" and "considerably accurate", as illustrated by Figure 5.8, p. 52. However, it is essential to note that one user found the tool's recognition "highly inaccurate". Even though the data indicates promising results regarding the tool's efficacy, one should factor the simplicity of proof 1 into the analysis.

Taking a look at the perceived recognition results of proof 2, shown in Figure 5.9, p. 53, we can see that the results are evenly spread out between "considerably inaccurate", "neither


Figure 5.6: Bar plot of the participants' perceived difficulty upon writing proof 1 on a tablet PC


Figure 5.7: Bar plot of the participants' perceived difficulty upon writing proof 2 on a tablet PC


Figure 5.8: Bar plot of the participants' perceived recognition accuracy upon writing proof 1 on a tablet PC
accurate nor inaccurate" and "considerably accurate" with four entries each. It stands to reason that individuals tend to find the system increasingly less accurate when writing a more complex proof. Nonetheless, only a third of the users consider the recognition's accuracy mediocre, indicating that most users will likely find the recognition results acceptable.

Regarding the participants' perception of the extent of needed corrections and rewrites needed, Figure 5.10 , p. 53 and Figure 5.11 , p. 54 show that there are little differences in both proofs, with most users believing they did required minimal rework. This suggests that the higher recognition inaccuracy of proof 2 may stem from challenges with recognition in proofs with content related to propositional logic rather than the proof's length or complexity.

Next, we asked the participants to compare the effort in terms of time consumption and complexity of manually inputting the $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ equivalent of both proofs with the effort of using the proposed system to obtain recognition. For this question, we provided the following selection of options:

1. Much more complex and time-consuming to use the proposed tool.
2. Slightly more complex and time-consuming to use the proposed tool.
3. Similar level of complexity and time required for both methods.
4. Slightly more complex and time-consuming to manually type the $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ equivalent.
5. Much more complex and time-consuming to manually type the $\mathrm{ET}_{\mathrm{E}} \mathrm{X}$ equivalent.

As we can observe, although the options are formulated more intricately, the question can still be mapped to the Likert scale. As such, considering the options an ascending numeration, we can


Figure 5.9: Bar plot of the participants' perceived recognition accuracy upon writing proof 2 on a tablet PC


Figure 5.10: Bar plot of the participants' perceived need for rewriting and correction upon writing proof 1 on a tablet PC


Figure 5.11: Bar plot of the participants' perceived need for rewriting and correction upon writing proof 2 on a tablet PC
visually represent the data as seen in Figure 5.12, p. 55 and Figure 5.13, p. 55, where a 1 represents a solid inclination to use the proposed tool and a 5 represents a strong inclination to type the $\mathrm{EAT}_{\mathrm{E}} \mathrm{X}$ equivalent manually.

Proof 1's IATEX equivalent is very similar to that which can be seen in Figure 5.1, p. 45. The LeTEX equivalent of proof 2 (see Figure 5.2, p. 45) on the other hand is as follows:

```
(p \wedge q) \vee (p \wedge r)
= \{ ... \}
((p \wedge q) \vee p) \wedge ((p \wedge q) \vee q)
= \{ ... \}
p \wedge ((p \wedge q) \vee r)
```

As we can observe, "lwedge" refers to the logical AND sign $\wedge$ and "lvee" refers to the logical OR sign $V$.

From the participants' feedback, we can see a split in the perceptions for the first proof. Half of the participants find it slightly more complex or of similar complexity to use the proposed tool. In contrast, the majority of the other half believe it is slightly more complex to type the LATEX equivalent. This is a significant insight considering that, as mentioned previously, the LATEX $^{2}$ equivalent of proof 1 is trivial to type with a keyboard. This suggests that a substantial segment of users value the convenience of a system like the one proposed, which provides immersion in the writing process while delivering robust, albeit imperfect, recognition results. This viewpoint becomes even more evident as we analyse the results for proof 2 , where almost half of the users reported that it would be much more complex to type the $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ equivalent.


Figure 5.12: Bar plot of the participants' perceived effort and complexity of manually inputting the IATEX equivalent versus the effort of using the proposed system to obtain recognition of proof 1 on a tablet PC. The questions are mapped to a Likert scale numerical format.


Figure 5.13: Bar plot of the participants' perceived effort and complexity of manually inputting the $\mathrm{LT}_{\mathrm{E}} \mathrm{X}$ equivalent versus the effort of using the proposed system to obtain recognition of proof 2 on a tablet PC. The questions are mapped to a Likert scale numerical format.

| Participant | Proof 1 (mins) | Proof 2 (mins) |
| :---: | ---: | ---: |
| P1 | 3.0 | 2.3 |
| P2 | 3.3 | 2.3 |
| P3 | 2.5 | 2.0 |
| P4 | 2.3 | 2.1 |
| P5 | 1.5 | 1.2 |
| P6 | 1.4 | 1.3 |
| P7 | 2.2 | 2.3 |
| P8 | 2.2 | 3.4 |
| P9 | 3.0 | 2.4 |
| P10 | 3.0 | 4.0 |
| P11 | 2.3 | 2.1 |
| P12 | 2.2 | 2.5 |

Table 5.1: Table comparing the time that each participant needed to write proofs 1 and 2 in minutes

### 5.6.2 Time

We measured the time each participant spent on each task, shown in Table 5.1. The statistics found in Table 5.2 show that, on average, participants required less than three minutes to write both proofs individually. Although outliers are inevitable in any data set, their impact is best represented through the boxplot of Figure 5.14, p. 57. Some users took nearly twice the average time to complete the second task. If we were to remove these extreme values from consideration, users transcribed the second proof faster than the first, even though it is more complex and lengthier. Additionally, it is worth noting that participants may take longer because of questions, other verbal remarks or periods of unfocused attention while writing the proofs. Lastly, considering that the standard deviation does not exceed 0.41 minutes, we can reasonably infer that most users would cluster around the mean.

| Statistic | Proof 1 | Proof 2 |
| :---: | ---: | ---: |
| Minimum | 1.4 | 1.2 |
| Maximum | 3.3 | 4.0 |
| Mean $(\mu)$ | 2.45 | 2.38 |
| Median | 2.3 | 2.3 |
| Standard Deviation $(\sigma)$ | 0.61 | 0.64 |
| Variance $\left(\sigma^{2}\right)$ | 0.37 | 0.41 |

Table 5.2: Table showing the statistical analysis for the time spent on proofs 1 and 2 in minutes


Figure 5.14: Box plot of the participants' required completion time for each proof

### 5.6.3 System Usability Scale

The System Usability Scale (SUS) [14] is a simple, industry-standard, technology-agnostic set of 10 questions on a 5-point Likert scale that is used to gauge the subjective impressions of users on several of the system's usability and user experience aspects. These are the ten questions that form the SUS and that were given to the participants:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

As we can observe, the SUS consists of alternatively positive and negative statements that can be calculated into a score with the following steps:

| Participant | SUS Score |
| :---: | ---: |
| P1 | 67.5 |
| P2 | 75.0 |
| P3 | 100.0 |
| P4 | 80.0 |
| P5 | 85.0 |
| P6 | 67.5 |
| P7 | 75.0 |
| P8 | 97.5 |
| P9 | 60.0 |
| P10 | 52.5 |
| P11 | 85.0 |
| P12 | 65.0 |
| Minimum | 52.5 |
| Maximum | 100.0 |

Table 5.3: Table comparing the System Usability Scale (SUS) total score for each participant, including minimum, maximum, mean, and median scores

1. For odd-numbered questions, subtract 1 from the user's response.
2. For even-numbered questions, subtract the user's response from 5 .
3. Sum up the scores from steps 1 and 2 .
4. Multiply this score by 2.5 .

The rationale behind this calculation is that responses indicative of satisfaction - whether they originate from questions posed positively or negatively - need to contribute equally to the overall score. By doing this, we can get a SUS score that represents the system subjectively, where a higher score indicates better-perceived usability. Table 5.3 indicates the SUS score that every participant gave to the system and Table 5.4, p. 59 shows the average score for each of the questions. We can infer that the participants generally believe that the system is easy to use, quick to learn, convenient and intuitive. On the other hand, it seems like the system is quite inconsistent and users are unlikely to use this system frequently. Overall, a score of $\sim 76.8$ can be considered quite good, falling above the scale's midpoint in terms of usability. To see the entries for each individual question, refer to appendix C .

### 5.6.4 Spearman's Rank-Order Correlation

Spearman's correlation coefficient [59] is a robust measure of correlation that does not make any assumptions about the given data upon which it is acting. It determines how well a monotonic function can describe a relationship between two variables without assuming anything relative to

| System Usability Scale (SUS) Question | Average Score |
| :--- | ---: |
| I think that I would like to use this system frequently. | 5.000 |
| I found the system unnecessarily complex. | 8.958 |
| I thought the system was easy to use. | 7.708 |
| I think I would need the support of a technical person to be able to use the system. | 8.542 |
| I found the various functions in this system were well integrated. | 7.083 |
| I thought there was too much inconsistency in this system. | 5.208 |
| I would imagine that most people would learn to use this system very quickly. | 9.167 |
| I found the system very cumbersome to use. | 8.125 |
| I felt very confident using the system. | 6.875 |
| I needed to learn a lot of things before I could get going with this system. | 9.167 |
| Total SUS Score $(\Sigma)$ | $\mathbf{7 6 . 8 3 3}$ |

Table 5.4: Table showing the average System Usability Scale (SUS) score for each individual question. Scores have been adjusted by a factor of 2.5 as per SUS calculation instructions
their nature. The calculated coefficient yields a value between -1 and 1 , where -1 indicates an entirely negative correlation, and 1 indicates an entirely positive correlation.

Pearson's correlation coefficient [52], a measure of linear correlation, also initially came to mind. However, that method is only partially appropriate, considering the collected data is in the form of a Likert scale. Although numerically coded, it may not have equally distant intervals between responses. For instance, a considerable agreement may be less distant from neutrality than to extreme agreement. Furthermore, Spearman's correlation coefficient is less sensitive to outliers than Pearson's, which also requires the datasets to be normally distributed. In contrast, Spearman's rank order does not make any assumptions about the distribution. These reasons lead to Spearman's test emerging as the most suitable.

Table 5.5, p. 60 shows some of the non-trivial correlations found. As it can be observed, we can argue with a $96.4 \%$ confidence that there is a relatively strong correlation between users exhibiting familiarity with handwriting recognition tools for mathematical expressions and those who found writing the second proof an easy task. The second correlation links users' familiarity with other tools with wanting to use the proposed system frequently at a confidence level of $99.0 \%$ with an even stronger correlation coefficient. Taken together, both of these correlations may indicate that users with experience in similar software would have an inclination towards using the system and little difficulty navigating it.

An analysis of the next set of correlations reveals some data on the user perception of the proposed tool. The correlation indicates that users who found the recognition to be accurate on one of the proofs also vouched for the accuracy of the other with $98.5 \%$ confidence. Additionally, a correlation with $99.5 \%$ confidence was found that relates users who find it more complex to manually type the $\mathrm{IAT}_{\mathrm{E}} \mathrm{X}$ equivalent of one proof and believe the same for the other. Furthermore, another correlation shows with $99.2 \%$ confidence that users who had no difficulty transcribing proof 2 consider using this system frequently. These three correlations imply that the system garners pos-

| Question 1 | Question 2 | Coefficient | Confidence |
| :--- | :--- | ---: | ---: |
| Familiarity with tools similar tools | Difficulty in proof 2 | -0.608 | $96.42 \%$ |
| Familiarity with tools similar tools | Frequent system use | 0.707 | $99.0 \%$ |
| Accuracy in recognition of proof 1 | Accurate recognition on proof 2 | 0.698 | $98.5 \%$ |
| Complexity of typing proof 1 | Complexity of typing proof 2 | 0.578 | $99.5 \%$ |
| Difficulty in proof 2 | Frequent system use | -0.719 | $99.2 \%$ |
| Confidence in system use | Accuracy in recognition of proof 2 | 0.643 | $97.5 \%$ |
| Confidence in system use | Complexity of typing proof 2 | -0.629 | $97.1 \%$ |

Table 5.5: Table showing the Spearman correlation and confidence levels between pairs of questions.
itive reception, at least for proofs that contain content related to arithmetic or propositional logic. These insights suggest a reliable and valuable tool for proof transcription.

The final pair of correlations show that people felt confident while using the system when the recognition of proof 2 was deemed accurate, and that confidence with the tool led to users finding using the tool less complex than manually typing the proof into LaTeX format.

### 5.6.5 Observations and Improvements

As mentioned, participants had the opportunity to make observations and suggest improvements at the end of the presented form, which we paraphrase next:

- One of the participants noted that the recognitions were not correct most of the time, referring to the fact that it might vary based on an individual's writing style.
- Another user noted that the software should improve due to inconsistency. They acknowledge, however, that it is faster to use the proposed tool than typing the proofs on a traditional PC in certain situations.
- Another participant observed that the software was considerably hard to use because of the tablet device used to write the proofs. They note that isolating the hardware difficulties from the software's efficiency is impossible. Nonetheless, they consider the software fairly intuitive if required to do an assessment.
- The last feedback refers to the fact that the system is interesting and easy to use, albeit needing refinement in the interpretation capabilities.


### 5.7 Threats to Validity

Although biases were already mentioned in Section 5.4, p. 46, there are some threats to the validity of the executed user study.

One of the ways the study may be compromised has to do with the participants' characteristics. The selection of the study's participants may not accurately reflect the tool's target users, which are likely to be scientists, researchers and educators. Even though the study sample was drawn from higher education, it may not accurately represent the visions of even more extensively educated individuals. It is also fair to say that there are biases related to age and gender, as almost all participants fall into the age range of $18-24$, and only $25 \%$ of the participants are of the female gender. Furthermore, involving approximately ten people in the study might not be an adequate sample size to generate decisive conclusions.

Additionally, it is important to consider that the participants may have been influenced by the Hawthorne effect [56]. This phenomenon describes the alteration of participants' behaviour when aware that they are being observed, potentially performing differently than they would in a more natural environment. This effect is also linked to the participants' inevitable awareness of the purpose of the study and knowing what is effectively being tested.

Participants could also fall victim to confirmation bias [65], wherein their interpretations are influenced by their pre-existing beliefs, potentially giving disproportionate considerations to alternative possibilities and preventing them from critically evaluating the tool's performance. For this reason, we request participants to evaluate themselves on their familiarity with other handwriting recognition systems before proceeding with the tasks.

Another component that may influence the results shown is the social desirability bias. This bias occurs when the participants answer questions in such a way that they believe to be favourably perceived by others, or, in this case, the creator of the user study, and consequently, potentially evaluate the tool more positively than they feel is right. To mitigate this, we encouraged participants to be honest and not observe the participants' answers when filling out the form.

Something to consider is how long the study goes on. Participants who have shorter attention spans or experience fatigue may see a decrease in performance. To prevent this effect, we chose short proofs for both tasks, as seen in Section 5.2, p. 44. However, it is equally crucial to consider that participants may also likely become more proficient with the tool over time, leading to better performance on later tasks, which is the case as explained in Section 5.6.2, p. 56, for example.

Finally, as referred by some participants both in the observations of the form (see Section 5.6.5, p. 60) and verbally, it is challenging to isolate the user experience on the software from the hardware used in the study. The Android tablet in question is considerably outdated, being a decade old and running an antiquated Android version 5, while the latest version is 13. Moreover, the screen is not very responsive to the pen's touch. Users sometimes expressed frustration when the tablet would not register their pen presses on the device's screen, causing a break in the interaction and immersion.

## Chapter 6

## Conclusions and Future Work

6.1 Summary ..... 62
6.2 Contributions ..... 63
6.3 Future Work ..... 63

This concluding chapter comprehensively reviews all of the work's accomplishments. Section 6.1 summarises what has been discussed in the dissertation. Section 6.2, p. 63 exposes this work's contributions to the field of online handwriting recognition systems. Finally, Section 6.3, p. 63 presents the areas for future improvement and the next steps in further research.

### 6.1 Summary

Chapter 1, p. 1 serves as the introduction for the thesis, presenting the current gap in online handwriting recognition tools for mathematical expressions that consider the structure of the calculational method and its surrounding context. It also underlines the relevance of tackling such a problem in a society where education in a digital format is increasing and the motivation and goals for solving such a challenge.

We dive into the current state of the art in Chapter 2, p. 5, starting by explaining essential core concepts crucial for comprehending the remaining work. Following this, we analyse the most important research in the field in question, as well as the relevant similar tools that serve as whiteboards and recognisers of mathematical expressions. Upon deliberation, Seshat - the opensource recogniser of mathematical expressions - was identified as the appropriate candidate for integration into an open-source whiteboard.

Chapter 3, p. 20 focuses on the system's desired behaviour, elucidating the underlying reasoning behind some of the decisions made and aiming to provide insight into the considerations made before the development process. It also details the interface's design and supported notation and highlights the attributes contributing to a modular structure.

The thesis's crux lies in Chapter 4, p. 25, which thoroughly dissects and analyses the applied solution. We start by giving an overview of the architectural and design patterns instrumental
in guiding the implementation and later describing each component's main functionalities and intricacies. Moreover, it explains some fundamental data structures and formats that store and share information through a server. Lastly, we detail some of the methods considered for line segmentation.

Chapter 5, p. 43 presents the performed user study validating the proposed tool, which showed promising results. It starts by defining the goals, detailing the guide, the questionnaire and the rationale behind each, and presenting the obtained results while weighing possible threats to the validity.

### 6.2 Contributions

It is fair to conclude that this dissertation addressed the paucity of tools that perform handwriting recognition designed for the Calculational Method, which was tackled by reviewing the state of the art, understanding its lack of software related to the Calculational Method, and consequently creating a system that merges a whiteboard interface with a server and Seshat. It also aimed to complete the work of Mendes [43, 44] by creating a suitable recogniser that can be used for a work that intends to address the issues of structured manipulation of mathematical expressions.

The resulting tool fills a void in educational technology, providing a solution that can be further developed for practical use for teaching, learning, and research activities related to proofs in the calculational style. It provides a much-needed platform for online handwriting recognition in an era where remote learning is becoming more frequent. The user study, which contributes to understanding users' preferences about handwriting recognition technology, shows that the tool is appropriate for writing proofs of multiple styles even though the recognition is only partially accurate occasionally. It performs well in terms of usability and is intuitive and effective, especially in the hands of scientists, researchers or educators.

### 6.3 Future Work

As for future work in online handwriting recognition systems of mathematical expressions, there is much room for improvement. A possible improvement involved the implementation of an algorithm that generates not a single recognition result but the N -best results instead. By implementing such a feature and displaying multiple results as an interactable suggestion, the user may be more inclined to use such a system. After all, when users find it easier to obtain the intended outcome, the system becomes more appealing.

Another feature that could significantly increase the system's usability would be the incorporation of free text recognition. Currently, the tool is solely trained to recognise mathematical expressions. However, proofs in the calculational style indispensably include natural language text often within their structure, in this case, inside the hints.

The most crucial enhancement, regrettably unfeasible in this dissertation's conditions, that would undoubtedly improve the tool's performance would be the retraining of Seshat's RNN
model. Given unrestricted access to the trained model, one could retrain it to take into account the inherent structure and symbols of the Calculational Method.

Finally, it is worthy of note that any software tool has several points which can be taken as future research directions regarding performance and accessibility, this one included.

## References

[1] Francisco Álvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and hidden markov models. Pattern Recognition Letters, 35:58-67, 2014.
[2] Francisco Álvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. An integrated grammarbased approach for mathematical expression recognition. Pattern Recognition, 51:135-147, 2016.
[3] Robert H Anderson. Syntax-directed recognition of hand-printed two-dimensional mathematics. In Symposium on interactive systems for experimental applied mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium, pages 436-459, 1967.
[4] Ahmad-Montaser Awal, Harold Mouchere, and Christian Viard-Gaudin. A global learning approach for an online handwritten mathematical expression recognition system. Pattern Recognition Letters, 35:68-77, 2014.
[5] Roland Backhouse. Mathematics and programming. a revolution in the art of effective reasoning. Inaugural Lecture, School of Computer Science and IT, University of Nottingham, 2001.
[6] Roland Backhouse, Richard Verhoeven, and Olaf Weber. Mathpad: A system for on-line preparation of mathematical documents. Software - Concepts and Tools, 18:80-, 011997.
[7] C. Bahlmann, B. Haasdonk, and H. Burkhardt. Online handwriting recognition with support vector machines - a kernel approach. In Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition, pages 49-54, 2002.
[8] Sunitha Basodi, Chunyan Ji, Haiping Zhang, and Yi Pan. Gradient amplification: An efficient way to train deep neural networks. Big Data Mining and Analytics, 3(3):196-207, 2020.
[9] Peter W Becker and Knud A Nielsen. Pattern recognition using dynamic pictorial information. IEEE Transactions on Systems, Man, and Cybernetics, SMC-2(3):434-437, 1972.
[10] Claude Berge. Hypergraphs: Combinatorics of Finite Sets. North-Holland, 1989.
[11] Ayan Kumar Bhunia, Abir Bhowmick, Ankan Kumar Bhunia, Aishik Konwer, Prithaj Banerjee, Partha Pratim Roy, and Umapada Pal. Handwriting trajectory recovery using end-to-end deep encoder-decoder network. In 2018 24th International Conference on Pattern Recognition (ICPR), pages 3639-3644. IEEE, 2018.
[12] Dorothea Blostein and Ann Grbavec. Recognition of mathematical notation. In Handbook of character recognition and document image analysis, pages 557-582. World Scientific, 1997.
[13] Norman M. Bradburn, Seymour Sudman, and Brian Wansink. Questionnaire design. Annual Review of Psychology, 50(1):263-294, 2004.
[14] John Brooke. Sus: A quick and dirty usability scale. Usability Evaluation in Industry, pages 189-194, 1986.
[15] Kam-Fai Chan and Dit-Yan Yeung. Mathematical expression recognition: a survey. International Journal on Document Analysis and Recognition, 3(1):3-15, 2000.
[16] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259, 2014.
[17] Noam Chomsky and George A Miller. Introduction to the formal analysis of natural languages. Journal of Symbolic Logic, 33(2), 1968.
[18] John Cocke, Daniel H. Younger, and Tadao Kasami. Cocke-younger-kasami algorithm (cyk algorithm). Journal of the ACM (JACM), 13(4):656-661, 1970.
[19] Andrew G. Cohn, Lei Ren, Amitabh Gupta, Jin Hyun You, David H. Kim, and Chengzhi Zhang. InkML: An XML-based standard for digital ink data. In Proceedings of the 8th International Conference on Document Analysis and Recognition (ICDAR), pages 628-633, 2005.
[20] Corinna Cortes and Vladimir Vapnik. Support-vector networks. Machine learning, 20:273297, 1995.
[21] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. IEEE transactions on information theory, 13(1):21-27, 1967.
[22] Hai Dai Nguyen, Anh Duc Le, and Masaki Nakagawa. Deep neural networks for recognizing online handwritten mathematical symbols. In 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), pages 121-125, 2015.
[23] Edsger W Dijkstra and Carel S Scholten. Predicate calculus and program semantics. Springer Science \& Business Media, 2012.
[24] Jeffrey L Elman. Finding structure in time. Cognitive science, 14(2):179-211, 1990.
[25] Claudie Faure and Zi Xiong Wang. Automatic perception of the structure of handwritten mathematical expressions. In Computer processing of handwriting, pages 337-361. World Scientific, 1990.
[26] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1):119-139, 1997.
[27] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric environment, 32(14-15):2627-2636, 1998.
[28] Trishita Ghosh, Shibaprasad Sen, Sk.Md. Obaidullah, K.C. Santosh, Kaushik Roy, and Umapada Pal. Advances in online handwritten recognition in the last decades. Computer Science Review, 46:100515, 2022.
[29] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd international conference on Machine learning, pages 369-376. ACM, 2006.
[30] Alex Graves and Alex Graves. Long short-term memory. Supervised sequence labelling with recurrent neural networks, pages 37-45, 2012.
[31] David Gries and Fred B Schneider. A logical approach to discrete math. Springer Science \& Business Media, 2013.
[32] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. Master's thesis, Institut fur Informatik, Technische Universitat, Munchen, 041991.
[33] R A Hought, J G Wilkes, and K W Bailey. Automated recognition of handprinted characters. IEEE Transactions on Systems, Man and Cybernetics, 5(11):767-777, 1975.
[34] Lei Hu and Richard Zanibbi. Segmenting handwritten math symbols using adaboost and multi-scale shape context features. In 2013 12th International Conference on Document Analysis and Recognition, pages 1180-1184. IEEE, 2013.
[35] Leslie K John, George Loewenstein, and Drazen Prelec. Measuring the prevalence of questionable research practices with incentives for truth telling. Psychological science, 23(5):524-532, 2012.
[36] Ray Kurzweil. Reading machine for the blind. Journal of the Franklin Institute, 298(6):219226, 1974.
[37] L Lamel, J Paquet, and Y Lecourtier. A system for handwritten text line recognition. In Proceedings of the Third International Conference on Document Analysis and Recognition, pages 777-782. IEEE, 1998.
[38] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks, 3361(10):1995, 1995.
[39] M Paul Lewis, Gary F Simons, and Charles D Fennig. Ethnologue: languages of the world, dallas, texas: Sil international. Online version: http://www. ethnologue. com, 12(12):2010, 2009.
[40] Rensis Likert. A Technique for the Measurement of Attitudes. Archives of Psychology, 1932.
[41] Hong Lin and Jun Tan. Application of deep learning in handwritten mathematical expressions recognition. In Yue Lu, Nicole Vincent, Pong Chi Yuen, Wei-Shi Zheng, Farida Cheriet, and Ching Y. Suen, editors, Pattern Recognition and Artificial Intelligence, pages 137-147, Cham, 2020. Springer International Publishing.
[42] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5:115-133, 1943.
[43] Alexandra Mendes. Structured editing of handwritten mathematics. PhD thesis, University of Nottingham, 2012.
[44] Alexandra Mendes, Roland Backhouse, and Joao F Ferreira. Structure editing of handwritten mathematics: Improving the computer support for the calculational method. In Proceedings of the Ninth ACM International Conference on Interactive Tabletops and Surfaces, pages 139-148, 2014.
[45] Erik G Miller and Paul A Viola. Ambiguity and constraint in mathematical expression recognition. In AAAI/IAAI, pages 784-791, 1998.
[46] Bennet B Murdock Jr. The serial position effect of free recall. Journal of Experimental Psychology, 64(5):482, 1962.
[47] George Nagy, Thomas A Nartker, and Stephen V Rice. Optical character recognition: An illustrated guide to the frontier. In Document recognition and retrieval VII, volume 3967, pages 58-69. SPIE, 1999.
[48] Cuong Tuan Nguyen, Thanh-Nghia Truong, Huy Quang Ung, and Masaki Nakagawa. Online handwritten mathematical symbol segmentation and recognition with bidirectional context. In 2020 17th International Conference on Frontiers in Handwriting Recognition (ICFHR), pages 355-360, 2020.
[49] Jakob Nielsen. Usability engineering. Elsevier, 1993.
[50] Ernesto Noya, Joan Andreu Sánchez, and José Miguel Benedí. Generation of hypergraphs from the n -best parsing of 2d-probabilistic context-free grammars for mathematical expression recognition. In 2020 25th International Conference on Pattern Recognition (ICPR), pages 5696-5703, 2021.
[51] Masayuki Okamoto and Akira Miyazawa. An experimental implementation of a document recognition system for papers containing mathematical expressions. In Structured Document Image Analysis, pages 36-53. Springer, 1992.
[52] Karl Pearson. Note on regression and inheritance in the case of two parents. Proceedings of the Royal Society of London, 58:240-242, 1895.
[53] Ian Pratt-Hartmann, Abdel Belaid, Christian Viard-Gaudin, Masaki Yoshida, and Richard Zanibbi. CROHME 2014: Competition on recognition of online handwritten mathematical expressions. In International Conference on Frontiers in Handwriting Recognition (ICFHR), pages 805-810, 2014.
[54] J. Ross Quinlan. Induction of decision trees. Machine learning, 1:81-106, 1986.
[55] L.R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2):257-286, 1989.
[56] F. J. Roethlisberger and W. J. Dickson. Management and the worker. Harvard University Press, 1939.
[57] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6):386, 1958.
[58] Steve Smithies, Kevin Novins, and James Arvo. A handwriting-based equation editor. In Graphics Interface, volume 99, pages 84-91, 1999.
[59] C. Spearman. The proof and measurement of association between two things. The American Journal of Psychology, 15(1):72-101, 1904.
[60] Johan AK Suykens and Joos Vandewalle. Least squares support vector machine classifiers. Neural processing letters, 9(3):293-300, 1999.
[61] Charles C. Tappert, Ching Y. Suen, and Toru Wakahara. The state of the art in online handwriting recognition. IEEE Transactions on pattern analysis and machine intelligence, 12(8):787-808, 1990.
[62] Antonetta JM Van Gasteren. On the shape of mathematical arguments, volume 445. Springer Science \& Business Media, 1990.
[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.
[64] L Wang and X Geng. A convolutional neural network-based framework for offline handwriting recognition. International Journal on Document Analysis and Recognition, 19(3):273285, 2016.
[65] Peter C Wason. On the failure to eliminate hypotheses in a conceptual task. Quarterly Journal of Experimental Psychology, 12(3):129-140, 1960.
[66] Oleg Yakovchuk, Anastasiia Cherneha, Dmytro Zhelezniakov, and Viktor Zaytsev. Methods for lines and matrices segmentation in rnn-based online handwriting mathematical expression recognition systems. In 2020 IEEE Third International Conference on Data Stream Mining and Processing (DSMP), pages 255-261, 2020.
[67] Jianshu Zhang, Jun Du, and Lirong Dai. A gru-based encoder-decoder approach with attention for online handwritten mathematical expression recognition. In 2017 14th IAPR International conference on document analysis and recognition (ICDAR), volume 1, pages 902-907. IEEE, 2017.
[68] Jianshu Zhang, Jun Du, and Lirong Dai. Track, attend, and parse (tap): An end-to-end framework for online handwritten mathematical expression recognition. IEEE Transactions on Multimedia, 21(1):221-233, 2018.
[69] Ting Zhang, Harold Mouchère, and Christian Viard-Gaudin. A tree-blstm-based recognition system for online handwritten mathematical expressions. Neural Computing and Applications, 32(9):4689-4708, 2020.

## Appendix A

## User Study Guide

This appendix contains the written guide containing the tasks presented to the voluntary participants of the user study.

# Tutorial and Experiments 

## Tablet PC Tool for Handwriting Recognition

## Introduction

Thank you for accepting this interview! This document will provide a tutorial and thorough instructions on handling the tool and solving the tasks you are asked for.

This study aims to obtain your opinion about a tablet PC tool for online handwriting recognition of mathematical proofs in the calculational style. This means you will write proofs in the tool, which will recognise the handwriting and display the LaTeX equivalent.

Here's what will be asked of you throughout this interview:

1. Go through a tutorial to get acquainted and comfortable with the tool
2. Write the proofs and save their recognition
3. Answer some questions in a form

Before starting the tutorial, please answer the first section of this form.

## Requirements

- Basic understanding of mathematics and propositional logic
- Internet connection

In the case of using a traditional PC for this study, a mouse is also recommended for a better user experience.

## Tutorial

Start by downloading and installing the tool with the instructions here.

## Warm-up

Let's try to write the expression $\boldsymbol{a}+\boldsymbol{b}=\boldsymbol{c}$ in the tool.
After that, change the expression to be $a+x=y$.
After writing the first expression, the screen should look something like the figure below.

## 

$$
a+b=c
$$

## 

## Instructions

To start drawing, ensure that the $\square$ tool is selected, press and hold left-click and then drag the mouse.

If you want to move the whiteboard, press the $\square$ tool and drag while left-clicking.

If you need to delete what you have written, you can delete all ink by pressing the
 button twice.

To undo or redo any strokes) you have written, use the buttons
 respectively.

To track the recognition of the current ink on the whiteboard, look at the top-right corner.

Finally, to download the current recognition, press the $\square$ button and choose somewhere for the file to be saved locally.

Once you're ready, press the $\square$ button to clear everything and start the tasks.

## Proof 1

Consider the following proof, which estimates a rough difference between 256 and 367, concluding that $\mathbf{3 6 7 - 2 5 6 < 2 0 0}$. Using the interface, write this proof giving enough spacing between each line.

Note that free text is not recognised in the tool. As such, substitute any free text with a blank. In this case, the hint that originally contains "arithmetic" should be " $=\{ \}$ " instead.
$\begin{array}{cc} & 367-256 \\ < & \{ \\ & 400-256 \\ < & \{ \\ & 400-200 \\ = & \{ \end{array} \quad$ arithmetic $\}$

Please answer section "Proof 1" of the form.

## Proof 2

Fill in the section for "Proof 2" on the form.

$$
\begin{aligned}
&(p \wedge q) \vee(p \wedge r) \\
&=\{\quad \text { distributivity of disjunction over conjunction (7.13) }\} \\
&=((p \wedge q) \vee p) \wedge((p \wedge q) \vee r) \\
&=\{\quad \text { absorption (see above) }\} \\
&=\left\{\begin{array}{l}
\text { distributivity of disjunction over conjunction } \\
\text { (symmetric version) }\}
\end{array}\right. \\
&= p \wedge(p \vee r) \wedge(q \vee r) \\
&\left\{\begin{array}{l}
\text { absorption }
\end{array}\right\} \\
& p \wedge(q \vee r) .
\end{aligned}
$$

## Appendix B

## User Study Questionnaire

Here we show the questions in a form given to the participants of the user study that accompanies the written guide of appendix A.

The main objective of this form is obtaining your opinion about a tablet PC tool for online handwriting recognition of mathematical proofs in the calculational style.

In the first sections of this form we gather demographic data and data on the environment you are using.

In the latter sections, related to the proofs we ask you to write, all gathered data is anonymous. This data may be used in presentations at conferences, academic events or similar events, and scientific publications.
Your participation is voluntary and you can withdraw at any time without any kind of penalty.

By selecting "Yes" below you are indicating that you agree to this data being processed, stored and used as described above and that you have read this consent form. If you select "No", you must end the trial immediately.YesNo

Age *

- $<18$18-24
〇 25-34
○ $35-44$
- 45-54
$\bigcirc>55$

Gender *

Male
$\bigcirc$ Female
Non-Binary
$\bigcirc$ Prefer not to say

How would you rate your digital literacy? *
Very weak
1
2

$\bigcirc$

Very strong

How familiar are you with tools for handwriting recognition of mathematical expressions (ex. MyScript, Mathpix, Wolfram Alpha)

|  | 1 | 2 | 3 | 4 | 5 |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Not familiar at all | $\bigcirc$ | $\bigcirc$ | $\bigcirc$ | $\bigcirc$ | $\bigcirc$ |  |
| Extremely familiar |  |  |  |  |  |  |

I'm currently using a... *tablet PC (ex. iPad / Android tablet).traditional PC.

I'm currently... *
using a digital pen.
$\bigcirc$ not using a digital pen.

What operating system (OS) are you currently using? *AndroidiOSWindowsChrome OSOther

What browser do you use most regularly? *

Google Chrome
$\bigcirc$ SafariMozilla FirefoxMicrosoft Edge
$\bigcirc$ OperaBraveUC BrowserYandex BrowserOther

Writing proof 1 was... *

|  | 1 | 2 | 3 | 4 | 5 |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| extremely easy | $\bigcirc$ | $\bigcirc$ | $\bigcirc$ | $\bigcirc$ | $\bigcirc$ |  |
| extremely difficult |  |  |  |  |  |  |

The recognition was... *
$\begin{array}{lllll}1 & 2 & 3 & 4 & 5\end{array}$
highly inaccurate
highly accurate

A lot of rewriting was needed. *

| 1 | 2 | 3 | 4 | 5 |
| :--- | :--- | :--- | :--- | :--- |

Strongly agree $\bigcirc \bigcirc \bigcirc \bigcirc$ Strongly disagree

Compare the complexity of manually typing the LaTeX equivalent of proof 1 with * using the proposed whiteboard recognition tool.

Which method is more challenging or requires greater effort?Much more complex and time-consuming to manually type the LaTeX equivalent.Slightly more complex and time-consuming to manually type the LaTeX equivalent.Similar level of complexity and time required for both methods.Slightly more complex and time-consuming to use the proposed tool.Much more complex and time-consuming to use the proposed tool.

Writing this proof with a traditional PC instead would be... *

| 1 | 2 | 3 | 4 | 5 |
| :--- | :--- | :--- | :--- | :--- |

much easier
much harder

Writing proof 2 was... *
$\begin{array}{lllll}1 & 2 & 3 & 4 & 5\end{array}$
extremely easy

$\bigcirc$ extemely difficult

The recognition was... *
highly inaccurate

highly accurate

A lot of rewriting was needed. *
Strongly agree
12
$\bigcirc \bigcirc$Strongly disagree

Compare the complexity of manually typing the LaTeX equivalent of proof 2 with * using the proposed whiteboard recognition tool.

Which method is more challenging or requires greater effort?Much more complex and time-consuming to manually type the LaTeX equivalent.Slightly more complex and time-consuming to manually type the LaTeX equivalent.Similar level of complexity and time required for both methods.Slightly more complex and time-consuming to use the proposed tool.Much more complex and time-consuming to use the proposed tool.

Writing this proof with a traditional PC instead would be... *
much easier

2
3
4
5$\bigcirc$much harder

I think that I would like to use this system frequently.

|  | 1 | 2 | 3 | 4 | 5 |  |
| :--- | :--- | :--- | :--- | :--- | :--- | :--- |
| Strongly disagree | $\bigcirc$ | $\bigcirc$ | $\bigcirc$ | $\bigcirc$ | $\bigcirc$ |  |
| Strongly agree |  |  |  |  |  |  |

I found the system unnecessarily complex. *
1
2
3
4
5
Strongly disagree
$\bigcirc$
$\bigcirc$
$\bigcirc \bigcirc$
$\bigcirc$
Strongly agree

I thought the system was easy to use. *
1
2
3
4
5
Strongly disagree
$\bigcirc$
$\bigcirc$
$\bigcirc$Strongly agree

For this option, please select "Strongly Agree" *

12
2
4
45

Strongly disagree

$\bigcirc$
$\square$ Strongly agree

I found the various functions in this system were well integrated.

| 1 | 2 | 3 | 4 | 5 |
| :--- | :--- | :--- | :--- | :--- |

Strongly disagree

$\bigcirc$ $\bigcirc$ $\bigcirc$ Strongly agree

I thought there was too much inconsistency in this system.
1
2
3
4
5
Strongly disagree$\bigcirc$
$\bigcirc$$\bigcirc$ Strongly agree

I would imagine that most people would learn to use this system very quickly.
Strongly disagree
1
$\bigcirc$
$\bigcirc$
$\bigcirc$
$\bigcirc$
$\bigcirc$
Strongly agree

I found the system very cumbersome to use.
$\begin{array}{lllll}1 & 2 & 3 & 4 & 5\end{array}$
Strongly disagree
O
$\bigcirc$
$\bigcirc$
$\bigcirc$
$\bigcirc$
Strongly agree

I felt very confident using the system.
Strongly disagree
12
3
4
5

Strongly agree

I needed to learn a lot of things before I could get going with this system.
Strongly disagree
1 2
34
5
$\bigcirc \bigcirc \bigcirc$ Strongly agree

Observations and improvement suggestions

A sua resposta

## Appendix C

## System Usability Scale Questionnaire Results

In this appendix we show the results of the each individual question part of the System Usability Scale.

I think that I would like to use this system frequently.
12 respostas


I found the system unnecessarily complex.
12 respostas


I thought the system was easy to use.
12 respostas


I think I would need the support of a technical person to be able to use the system.
12 respostas


I found the various functions in this system were well integrated.
12 respostas


I thought there was too much inconsistency in this system.
12 respostas


I would imagine that most people would learn to use this system very quickly.
12 respostas


I found the system very cumbersome to use.
12 respostas


I felt very confident using the system.
12 respostas


I needed to learn a lot of things before I could get going with this system.
12 respostas



[^0]:    ${ }^{1}$ End-to-end frameworks refer to methods in which the processing is all contained in a black box, receiving original data as input and processed, final data as output

[^1]:    ${ }^{2}$ https://openboard.ch/
    $3_{\text {https:/ /excalidraw.com/ }}$

[^2]:    ${ }^{4}$ https://github.com/xournalpp/xournalpp
    5https://github.com/OXOYO/XBoard

[^3]:    ${ }^{6}$ https://github.com/cracker0dks/whiteboard
    ${ }^{7}$ https://www.myscript.com/calculator/
    $8_{\text {https://www.nebo.app/ }}$

[^4]:    ${ }^{9}$ https://www.texthelp.com/en-gb/products/equatio/
    10 https://www.microsoft.com/en-us/microsoft-365/word
    ${ }^{11}$ https://docs.google.com/

[^5]:    12https://mathpix.com/
    13https://github.com/phatware/WritePadSDK

[^6]:    ${ }^{14}$ https://github.com/samkit-jain/Handwriting-Recognition
    15https://github.com/Glyphoid/math-handwriting-lib
    16https://github.com/falvaro/seshat

[^7]:    ${ }^{1}$ https://www.microsoft.com/en-us/microsoft-365/word

[^8]:    ${ }^{1}$ https://github.com/nunores/Tablet-PC-Tool-for-Handwriting-Recognition
    $2^{2}$ https://github.com/cracker0dks/whiteboard
    $3^{3}$ https://expressjs.com/
    $4_{\text {https://github.com/falvaro/seshat }}$

[^9]:    $6_{\text {https: }}$ //www.w3.org/TR/MathML3/

[^10]:    7 https://www.mathjax.org/
    8https://katex.org/

[^11]:    ${ }^{9}$ https://sourceforge.net/p/rnnl/wiki/Home/
    10 https://xerces.apache.org/xerces-c/

