UNIVERSITAT POLITÈCNICA DE VALÈNCIA
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Mathematical Expression Recognition based on Probabilistic Grammars

By
Francisco Álvaro Muñoz

written under the direction of
Dr. Joan Andreu Sánchez and Prof. José Miguel Benedí

April 10, 2015

Mathematical Expression Recognition based on Probabilistic Grammars

By

Francisco Álvaro Muñoz

A thesis dissertation submitted to the
Departamento de Sistemas Informáticos y Computación,
Universitat Politècnica de València
in partial fulfillment of the requirements
for the degree of
Doctor en Informática
written under the direction of
Dr. Joan Andreu Sánchez and Prof. José Miguel Benedí
Valencia, Spain, April 10, 2015

Work supported by the European Union 7th Framework Program under the tranScriptorium project (Ref: 600707), and by the EC (FEDER/FSE) and the Spanish MEC/MICINN under the MIPRCV "Consolider Ingenio 2010" program (CSD2007-00018) and the FPU scholarship AP2009-4363.

A mis padres Paco y Magdalena

Copyright Note

I hereby declare that this dissertation describes my own original work developed under the direction of my two supervisors Dr. Joan Andreu Sánchez and Prof. José Miguel Benedí. Further, I have acknowledged all sources used and have cited these in the bibliography section.

Additionally, we have collaborated with other researchers in order to tackle some of the tasks reported in this thesis:

- The work regarding spatial relationships classification and shape descriptors (Section 4.2.2) was made in collaboration with Dr. Richard Zanibbi from the Rochester Institute of Technology, USA.
- The application of 2D-PCFG to layout analysis of structured documents (Section 8.4) results from the collaboration with Francisco Cruz and Dr. Oriol Ramos Terrades, from the Universitat Autònoma de Barcelona, Spain. In this task, they were experts on the image analysis of the document images and PGMs, while we focused on the structural parsing process of the documents.
- The μCAPTCHA proposal (Section 8.3) was jointly developed with Dr. Luis A. Leiva, colleague from the Universitat Politècnica de València. He is also responsible for the LaTeX transcription and information retrieval application presented in Section 8.2.

Finally, we have used public software and libraries in some of the developments of this thesis. We have explicitly acknowledged them in the corresponding sections of this document.

Acknowledgements / Agradecimientos

Aunque estas líneas aparecen al principio de esta tesis, son las últimas que escribo. Con el documento terminado, echo la vista atrás y veo el resultado de más de cinco años de trabajo dedicados a resolver un problema. Muchas personas han formado parte de mi vida durante este periodo de tiempo y han influido de un modo u otro, y aquí quiero agradecérselo.

En primer lugar me gustaría dar las gracias a mis directores Joan Andreu y José Miguel por haberme dado la oportunidad de hacer el doctorado con ellos. Hemos trabajado duro durante mucho tiempo, y esta tesis es el fruto de numerosas reuniones, discusiones, correos, revisiones... que representan una parte intangible a la par que esencial de este documento.

I would like to thank Richard Zanibbi and the DPRL lab for their warm welcome during my research stay at the cold Rochester Institute of Technology. I spent an unforgettable period of time over there, where I had the chance to discuss this specific research field, and also, I met many nice people with whom I am still in touch. Attending conferences was also a great experience that allowed me to meet people (kind as well as smart) from all over the world and share interesting ideas with them.

Many thanks to the reviewers and the board committee for accepting and making the effort of evaluating this thesis. Their comments and suggestions have positively contributed to the final state of this document and it is a great pleasure to receive feedback from people with such knowledge and research experience.

El PRHLT ha sido mi casa durante más de cinco años, un grupo lleno de gente más que brillante con unos niveles de exigencia que hacen que todos nos esforcemos por ser mejores. El ambiente de trabajo estos años ha sido extraordinario, muchos de nosotros hemos pasado de compañeros a ser grandes amigos, y la verdad es que he disfrutado mucho yendo a trabajar cada día. Echaré de menos conversaciones capaces de combinar la ciencia más rigurosa con el humor más fino, tardes acelerando para llegar a tiempo a un *deadline* y algún que otro frenazo.

Cómo no, he de agradecer a las personas que me han apoyado (y soportado) durante este periodo de tiempo. Entre los desafortunados que han estado conmigo en los buenos y en los malos momentos, están muy especialmente la peña de Aras, un grupo que germinó en un lugar remoto de la serranía combinando a gente única con la que he tenido la suerte de haber coincidido. Ni siquiera soy capaz de recordar cuando nos conocimos, y aunque nos hayamos repartido un poco por el mundo, cuando nos vemos parece que no haya pasado el tiempo.

Por último, a mi familia, a mis padres y a mi hermana. Por estar siempre ahí, confiar en mi y apoyarnos unos a otros en todo lo que hacemos y nos ha tocado vivir. Aunque no lo digamos a menudo, lo demostramos cada día.

Gracias a todos.

Valencia, 10 de abril de 2015.

Abstract / Resumen / Resum

Mathematical notation is well-known and used all over the world. Humankind has evolved from simple methods representing countings to current well-defined math notation able to account for complex problems. Furthermore, mathematical expressions constitute a universal language in scientific fields, and many information resources containing mathematics have been created during the last decades. However, in order to efficiently access all that information, scientific documents have to be digitized or produced directly in electronic formats.

Although most people is able to understand and produce mathematical information, introducing math expressions into electronic devices requires learning specific notations or using editors. Automatic recognition of mathematical expressions aims at filling this gap between the knowledge of a person and the input accepted by computers. This way, printed documents containing math expressions could be automatically digitized, and handwriting could be used for direct input of math notation into electronic devices.

This thesis is devoted to develop an approach for mathematical expression recognition. In this document we propose an approach for recognizing any type of mathematical expression (printed or handwritten) based on probabilistic grammars. In order to do so, we develop the formal statistical framework such that derives several probability distributions. Along the document, we deal with the definition and estimation of all these probabilistic sources of information. Finally, we define the parsing algorithm that globally computes the most probable mathematical expression for a given input according to the statistical framework.

An important point in this study is to provide objective performance evaluation and report results using public data and standard metrics. We inspected the problems of automatic evaluation in this field and looked for the best solutions. We also report several experiments using public databases and we participated in several international competitions. Furthermore, we have released most of the software developed in this thesis as open source.

We also explore some of the applications of mathematical expression recog-

nition. In addition to the direct applications of transcription and digitization, we report two important proposals. First, we developed μCAPTCHA, a method to tell humans and computers apart by means of math handwriting input, which represents a novel application of math expression recognition. Second, we tackled the problem of layout analysis of structured documents using the statistical framework developed in this thesis, because both are two-dimensional problems that can be modeled with probabilistic grammars.

The approach developed in this thesis for mathematical expression recognition has obtained good results at different levels. It has produced several scientific publications in international conferences and journals, and has been awarded in international competitions.

La notación matemática es bien conocida y se utiliza en todo el mundo. La humanidad ha evolucionado desde simples métodos para representar cuentas hasta la notación formal actual capaz de modelar problemas complejos. Además, las expresiones matemáticas constituyen un idioma universal en el mundo científico, y se han creado muchos recursos que contienen matemáticas durante las últimas décadas. Sin embargo, para acceder de forma eficiente a toda esa información, los documentos científicos han de ser digitalizados o producidos directamente en formatos electrónicos.

Aunque la mayoría de personas es capaz de entender y producir información matemática, introducir expresiones matemáticas en dispositivos electrónicos requiere aprender notaciones especiales o usar editores. El reconocimiento automático de expresiones matemáticas tiene como objetivo llenar ese espacio existente entre el conocimiento de una persona y la entrada que aceptan los ordenadores. De este modo, documentos impresos que contienen fórmulas podrían digitalizarse automáticamente, y la escritura se podría utilizar para introducir directamente notación matemática en dispositivos electrónicos.

Esta tesis está centrada en desarrollar un método para reconocer expresiones matemáticas. En este documento proponemos un método para reconocer cualquier tipo de fórmula (impresa o manuscrita) basado en gramáticas probabilísticas. Para ello, desarrollamos el marco estadístico formal que deriva varias distribuciones de probabilidad. A lo largo del documento, abordamos la definición y estimación de todas estas fuentes de información probabilística. Finalmente, definimos el algoritmo que, dada cierta entrada, calcula globalmente la expresión matemática más probable de acuerdo al marco estadístico.

Un aspecto importante de este trabajo es proporcionar una evaluación objetiva de los resultados y presentarlos usando datos públicos y medidas estándar. Por ello, estudiamos los problemas de la evaluación automática en este campo y buscamos las mejores soluciones. Asimismo, presentamos diversos experi-

mentos usando bases de datos públicas y hemos participado en varias competiciones internacionales. Además, hemos publicado como código abierto la mayoría del software desarrollado en esta tesis.

También hemos explorado algunas de las aplicaciones del reconocimiento de expresiones matemáticas. Además de las aplicaciones directas de transcripción y digitalización, presentamos dos propuestas importantes. En primer lugar, desarrollamos μCAPTCHA*, un método para discriminar entre humanos y ordenadores mediante la escritura de expresiones matemáticas, el cual representa una novedosa aplicación del reconocimiento de fórmulas. En segundo lugar, abordamos el problema de detectar y segmentar la estructura de documentos utilizando el marco estadístico formal desarrollado en esta tesis, dado que ambos son problemas bidimensionales que pueden modelarse con gramáticas probabilísticas.*

El método desarrollado en esta tesis para reconocer expresiones matemáticas ha obtenido buenos resultados a diferentes niveles. Este trabajo ha producido varias publicaciones en conferencias internacionales y revistas, y ha sido premiado en competiciones internacionales.

La notació matemàtica és ben coneguda i s'utilitza a tot el món. La humanitat ha evolucionat des de simples mètodes per representar comptes fins a la notació formal actual capaç de modelar problemes complexos. A més, les expressions matemàtiques constitueixen un idioma universal al món científic, i s'han creat molts recursos que contenen matemàtiques durant les últimes dècades. No obstant això, per accedir de forma eficient a tota aquesta informació, els documents científics han de ser digitalitzats o produïts directament en formats electrònics.

Encara que la majoria de persones és capaç d'entendre i produir informació matemàtica, introduir expressions matemàtiques en dispositius electrònics requereix aprendre notacions especials o usar editors. El reconeixement automàtic d'expressions matemàtiques té per objectiu omplir aquest espai existent entre el coneixement d'una persona i l'entrada que accepten els ordinadors. D'aquesta manera, documents impresos que contenen fórmules podrien digitalitzar-se automàticament, i l'escriptura es podria utilitzar per introduir directament notació matemàtica en dispositius electrònics.

Aquesta tesi està centrada en desenvolupar un mètode per reconèixer expressions matemàtiques. En aquest document proposem un mètode per reconèixer qualsevol tipus de fórmula (impresa o manuscrita) basat en gramàtiques probabilístiques. Amb aquesta finalitat, desenvolupem el marc estadístic formal que deriva diverses distribucions de probabilitat. Al llarg del document, abordem la definició i estimació de totes aquestes fonts d'informació probabilística. Finalment,

definim l'algorisme que, donada certa entrada, calcula globalment l'expressió matemàtica més probable d'acord al marc estadístic.

Un aspecte important d'aquest treball és proporcionar una avaluació objectiva dels resultats i presentar-los usant dades públiques i mesures estàndard. Per això, estudiem els problemes de l'avaluació automàtica en aquest camp i busquem les millors solucions. Així mateix, presentem diversos experiments usant bases de dades públiques i hem participat en diverses competicions internacionals. A més, hem publicat com a codi obert la majoria del software desenvolupat en aquesta tesi.

També hem explorat algunes de les aplicacions del reconeixement d'expressions matemàtiques. A més de les aplicacions directes de transcripció i digitalització, presentem dues propostes importants. En primer lloc, desenvolupem μCAPTCHA, un mètode per discriminar entre humans i ordinadors mitjançant l'escriptura d'expressions matemàtiques, el qual representa una nova aplicació del reconeixement de fórmules. En segon lloc, abordem el problema de detectar i segmentar l'estructura de documents utilitzant el marc estadístic formal desenvolupat en aquesta tesi, donat que ambdós són problemes bidimensionals que poden modelar-se amb gramàtiques probabilístiques.

El mètode desenvolupat en aquesta tesi per reconèixer expressions matemàtiques ha obtingut bons resultats a diferents nivells. Aquest treball ha produït diverses publicacions en conferències internacionals i revistes, i ha sigut premiat en competicions internacionals.

Contents

Chapter

1 Introduction

Mathematical notation is a well-known language that has been used all over the world since hundreds of years ago. Despite the great number of cultures, languages and even different writing scripts, mathematical expressions constitute a universal language in many fields. During last century and, especially, since the development of Internet, digital information represents the best resource in order to access and share data. Therefore, it is necessary to digitize documents and to input mathematical expressions directly into computers.

Although most people know how to read or write a mathematical expression, introducing them in a computer device usually requires learning specific notations or how to use a certain editor. Mathematical expression recognition intends to fill this gap between the knowledge of a person and the language computers understand.

This chapter presents the motivation for recognizing mathematical expressions and its applications in Section 1.1. Then, in Section 1.2 we introduce the types of mathematical expressions and the main problems that a recognition system hast to deal with. Finally, we describe the goals of this thesis in Section 1.3 and the structure followed to accomplish them is presented in Section 1.4.

Chapter Outline

1.1 Motivation

Mathematical notation constitutes an essential source of information in many fields. Thousands of years ago humankind started to represent countings using very primitive systems. Several civilizations created more and more complex notations in order to account for more advanced data, like geometric calculations or astronomical observations. The continuous developments of mathematics make possible that we can access today to the knowledge acquired for centuries.

Nowadays, there is an advanced well-known notation used all over the world. The amount of scientific documents is extraordinary, and the great scientific community is continuously producing more knowledge that has to be represented. Having the information digitalized is crucial in order to make feasible searching and accessing a set of relevant documents. Also, scientists, students and teachers access to math information and work with computers. However, introducing math notation into an electronic device requires learning notations like LaTeX and MathML, or using graphic editors.

Recognition of mathematical expressions aims at developing systems able to automatically understand math notation provided by humans as printed or handwritten expressions. This is important for automatic document digitization containing formulae, where scanned expressions can be recognized and there is a large amount of resources to digitize. Furthermore, it is very useful being able to input mathematical information into computers by taking a photo of an equation, or handwriting it directly into a tactile device or pen-based interface. Once a formula has been introduced into a computer, it can be handled since there are multiple applications.

Information retrieval is another interesting application of recognizing mathematical notation, such that a set of relevant documents can be obtained using a mathematical expression as a query. Finally, these techniques can also be used for accessibility of disable people, for example, a printed or handwritten mathematical expression can be translated into Braille notation or read using text-to-speech tools.

1.2 Mathematical Expression Recognition

Mathematical expression recognition is a pattern recognition problem and its goal is to obtain the math expression encoded in a given input sample. In this field we distinguish different types of expressions that require specific treatment. In this section, we first describe the taxonomy of mathematical

expressions considered in this problem. Afterwards, we state the main tasks that a math recognition system has to deal with.

First, a math expression can be either printed or handwritten. Printed formulae are commonly easier to recognize than handwritten expressions because they tend to be more regular. Handwriting introduces more variability in the shape of the symbols and the relations between them. Also, there are many different writers and writing styles, thus, handwritten mathematical expression recognition is more challenging. Figures 1.1 and 1.2 show an example of the printed and handwritten version of the same mathematical expression.

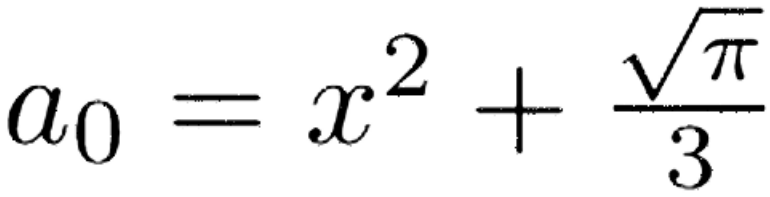


Figure 1.1: Example of printed mathematical expression.

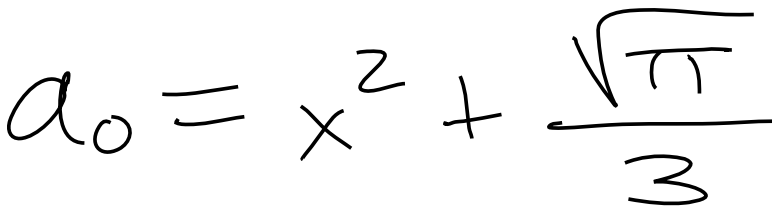


Figure 1.2: Example of handwritten mathematical expression.

Regarding the input representation, we consider the problem to be *offline* if the math expression is represented as an image, i.e. a matrix of pixels. On the other hand, a mathematical expression is considered to be *online* when it has been introduced using a device such that we have the temporal information of the writing, i.e. the input is a sequence of points. The representation of mathematical expressions is based on different primitives depending on the type of expression. Offline expressions are usually based on connected components.

Definition 1.2.1. A *connected component* of an image is a set of adjacent foreground pixels, where pairs of pixels are connected such that are neighbors in an 8-connected sense.

The primitives for representing online math expressions are usually strokes.

Definition 1.2.2. A *stroke* is the sequence of points drawn since the pen touches the surface until the user lifts the pen from the surface.

These definitions of primitives can be seen on the examples of Figures 1.1 and 1.2. For instance, in the printed expression of Figure 1.1, symbols π and $+$ are made up of one connected component, and symbol $=$ is composed of two connected components. If the handwritten expression of Figure 1.2 was online, symbol π would be composed of three strokes, symbol $+$ would be composed of two strokes and number 0 would be drawn using just one stroke. But if the handwritten expression of Figure 1.2 was offline, symbol π would be composed of two connected components, and symbol $+$ and number 0 would be made up of one connected component.

According to these differences, the problem of recognizing mathematical expressions has three possible scenarios: offline printed math expression recognition, and both online and offline handwritten math expression recognition.

Automatic recognition of math notation is traditionally divided into three problems [Chan and Yeung, 2000; Zanibbi and Blostein, 2012]: symbol segmentation, symbol recognition and structural analysis. The issues related to each of these problems are detailed below.

1.2.1 Symbol Segmentation

Symbol segmentation is the problem of determining what parts of the input expression form a mathematical symbol. Depending on the type of expression, online or offline, there are different issues to cope with.

Segmentation of offline mathematical expressions is usually based on computing the connected components of the image. Figure 1.3 shows examples of the problems related to segmentation of offline mathematical symbols. First, many symbols are made up of more than one connected component by definition and have to be grouped (Figure 1.3a). A difficult problem in offline segmentation is when the connected components of two different symbols are touching and have to be split (Figure 1.3b). Also, some symbols can be broken into several components due to image degradation (Figure 1.3c). Finally, images can contain noise that produces additional components that do not form any symbol and should be grouped or ignored (Figure 1.3d).

(a) (b) (c) (d)

Figure 1.3: Symbol segmentation problems in offline mathematical expression recognition based on connected components.

Segmentation of online expressions is commonly based on strokes, although symbol segmentation could also be based on connected components where it would have the same problems than in offline segmentation. Stroke-based segmentation can handle the problem of touching symbols provided that two touching symbols are written in different strokes. An equivalent problem would appear if two symbols share a single stroke.

Mathematical symbols can be made up of one or more strokes. For instance, in offline segmentation a plus sign (+) is detected as a connected component, but in online segmentation it has two strokes that must be merged. Figure 1.2 shows several multi-stroke symbols whose strokes must be grouped: i, π, $+$ and $=$. Finally, although is less common than noise in images, small strokes can be introduced by the user that do not belong to any symbol of the mathematical expression and should be discarded.

Context information is important to determine the segmentation of an input expression. In Figure 1.4(a) we can see several segmentation hypotheses that could be reasonable if context is not taken into account. For instance, the plus-minus sign ($\pm$) could be split into a plus sign and a horizontal line, or the fraction line and the top stroke of the number five could be merged as an equals sign [Awal et al., 2009]. Figure 1.4(b) shows ambiguities due to handwriting production, such that the expression could have several valid interpretations with alternative segmentations: $\boxed{1-1<x}$, $\boxed{1-kx}$ or $\boxed{H<x}$.

1.2.2 Symbol Recognition

Mathematical symbol recognition aims to identify the symbol encoded by a given hypothesis. Commonly, in offline recognition a hypothesis is an image, and in online recognition a symbol hypothesis is a set of strokes. There are many sets of symbols in mathematical notation: the Latin alphabet ($a-z$, $A-Z$), the Greek alphabet (α, β, γ, π, $\sum$, ...), numbers ($0-9$), operators ($+$, $-$, $/$, $\int$, $\sqrt{}$, ...) and more (∞, $\rightarrow$, $\forall$, $\{$, $\}$, ...).

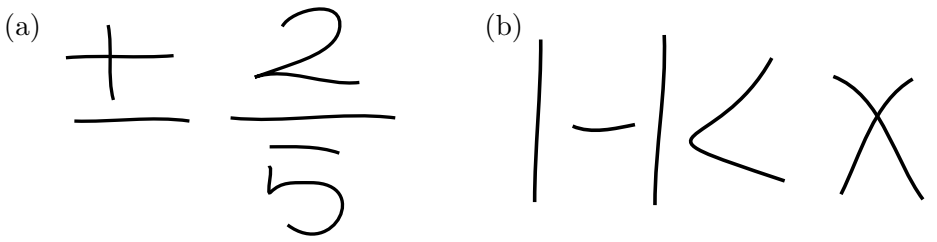


Figure 1.4: Handwritten mathematical expressions showing several examples of ambiguities in symbol segmentation.

Some of the symbols in mathematics are very similar, and there are even symbols that are represented by the same shape, for instance, the number 0 and the letter o, or the letter x and the operator $\times$. Figure 1.5 shows two examples of handwritten symbol hypotheses and some of their possible interpretations. Context information of the mathematical expression can help to solve the ambiguities in symbol recognition and determine the correct symbol class. We can see an example in Fig . 1.6 where the same shape represents a letter (x) in the upper expression and an Cartesian product operator ($\times$) in the lower expression.

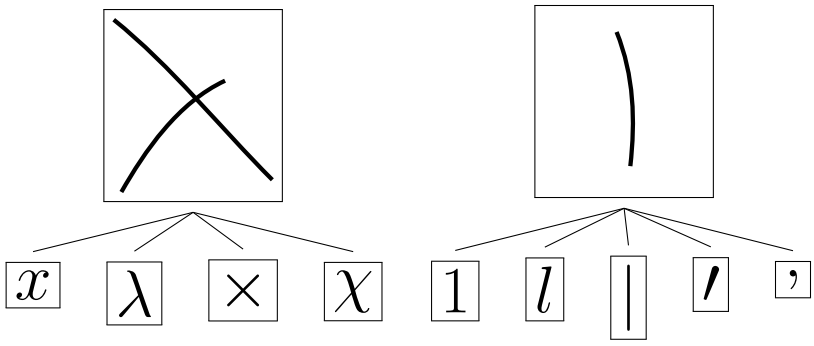


Figure 1.5: Recognition of mathematical symbols can be hard without context, examples of ambiguity in classification.

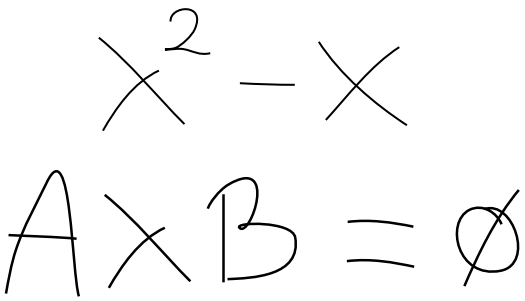


Figure 1.6: Example of symbol classification depending on the context in the mathematical expression. The same symbol shape is classified as a letter in top expression ($x^2 - x$) and as a product operator between sets in bottom expression ($A \times B = \emptyset$).

1.2.3 Structural Analysis

A mathematical expression is made up of symbols such that there are different relationships between them. Given a set of symbols, the final step in mathematical expression recognition is the detection of the structure that relates them and represents the recognized expression.

The structural analysis of equations requires determining two-dimensional relationships between symbols or sets of symbols. In Figures 1.1 and 1.2 we can see examples of the most common relations: *subscript* between symbols a and 0; *superscript* between symbols x and 2; *below* between the elements of the fraction; *inside* in the square root; and *right* relationship for the horizontal concatenation of the elements in the expression. There are other relations like radicals ($\sqrt[x]{\ }$), left scripts (${}_b^a x$) or the complex structure of matrices.

Relations between symbols can be ambiguous in several situations, such that detecting the correct relationship might require knowing the symbols or rely on language models. Figure 1.7 shows an example where the relationship between two symbols in a mathematical expression requires taking into account the entire expression.

Normally, the structure of a mathematical expression is represented as a tree. Figure 1.8 shows the same expression encoded by three types of trees. In relational trees, the mathematical symbols are the leaves of the tree and the internal nodes represent the relations, while in symbol layout trees each node is a symbol and edges indicate the relationships. In operator trees, leaves represent symbols and each node contains the operation that computes the expression bottom-up.

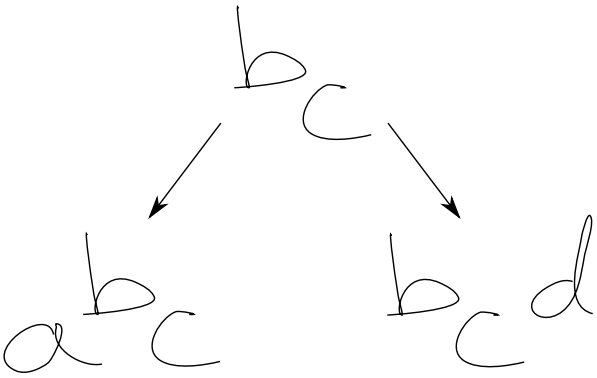


Figure 1.7: Example of subscript and superscript relationships that cannot be determined locally [Chan and Yeung, 2000].

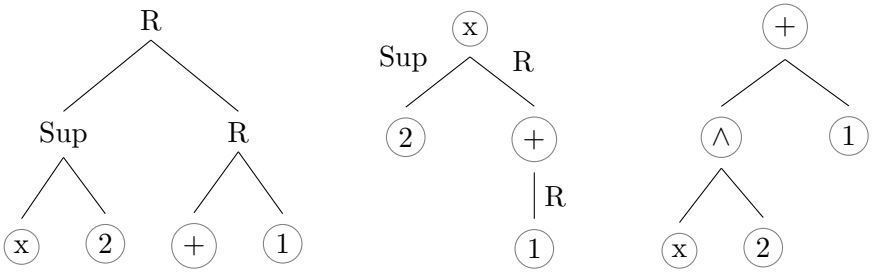


Figure 1.8: Example of tree representation of the expression $\boxed{x^2+1}$. Left to right: relational tree, symbol layout tree, and operator tree.

Finally, structural analysis is important in order to determine the correct segmentation and the identity of the recognized mathematical symbols. Some symbols can only be correctly classified if their spatial relationships are taken into account. For example, a horizontal line can represent a minus operator, a fraction bar or can be part of a symbol (e.g. $=$, $\leq$ or $\pm$). Context information is crucial to solve these ambiguities, as we can see in the example of Figure 1.4.

1.3 Scientific Goals

As previously described, there are several scenarios for mathematical expression recognition. An expression can either be printed or handwritten, and handwritten formulae can be, in turn, online or offline. Regardless of the type of math expression, three main problems must be solved: symbol segmentation, symbol recognition and structural analysis. These problems are closely related since symbol segmentation and symbol recognition depend on the structure of the expression, and vice versa.

This thesis is devoted to develop an integrated approach for mathematical expression recognition such that symbol segmentation, symbol recognition and structural analysis are simultaneously optimized. We want to define a formal framework based on parsing probabilistic grammars that can be generalized for any type of mathematical expression. Also, the model should virtually accept any input expression without constraints. Furthermore, we plan to deal with the estimation of the different probability distributions that yield from the recognition framework.

In this research field, despite many publications have been proposed, there are several issues that make comparison difficult. As a pattern recognition problem, we consider an important goal to evaluate the proposed methods using standard metrics and to report results using public corpora. This will let us to find the best features and classifiers for each task and provide comparable results. Also, we aim to develop the software required to implement the approach presented in this thesis and release it as open-source. We hope that this will help the progress in mathematical expression recognition.

Finally, we aim to explore some of the applications of the developments of this thesis. First, the range of applications of mathematical expression recognition as described at the beginning of this chapter. Second, the solutions and probabilistic models presented in this thesis could be applied to other tasks.

1.4 Document Structure

This Chapter 1 introduces the problem of recognizing mathematical expressions and defines the different goals of this thesis. In order to achieve them, the document is organized as follows:

Chapter 2 first reviews the related work on mathematical expression recognition. Then, we define the formal framework of the integrated approach

for recognizing mathematical notation developed in this thesis. The formulation of the model can account for any type of expression: printed expressions and both online and offline handwritten expressions.

Chapter 3 describes the probabilistic models that form the symbol likelihood probability derived in the statistical framework. The models, features and classifiers of this chapter are described for both handwritten and printed mathematical symbols.

Chapter 4 details the calculation of the probability that accounts for the structure of a mathematical expression. This chapter describes the computation of the spatial relationships between symbols and subexpressions, and the estimation of the probabilistic grammar.

Chapter 5 presents the parsing algorithm that computes the recognized mathematical expression for a given input such that the probability defined in the statistical framework is maximized. Furthermore, the properties of the algorithm are analyzed and some practical issues are discussed.

Chapter 6 introduces the problems of automatic performance evaluation in mathematical expression recognition. Then, we present different metrics that have been proposed in order to provide objective automatic assessment of recognition results.

Chapter 7 reports the experimentation carried out in order to evaluate at different levels the approach for math expression recognition developed in this thesis. The experiments are performed on both online handwritten math expressions and offline printed math expressions.

Chapter 8 presents several applications of the developments of this thesis. First, direct applications of mathematical expression recognition. Then, we detail the application of math expression recognition and probabilistic parsing to two different tasks.

Chapter 9 summarizes the contributions of this thesis and presents the publications resulted from this work.

Chapter

2 Mathematical Expression Recognition

Automatic recognition of mathematical expressions is a challenging problem that requires tackling ambiguities at several levels. On the one hand, the segmentation of the input in mathematical symbols and their classification. On the other hand, the detection of the two-dimensional structure that relates the symbols and represents the math expression. All these problems are closely related since symbol segmentation and recognition are influenced by the structure of the expression, while the structure strongly depends on the symbols recognized. For these reasons, we present an approach that integrates several stochastic sources of information and is able to globally determine the most likely expression. This way, symbol segmentation, symbol recognition and structural analysis are simultaneously optimized.

As previously introduced, there are different types of expressions. We present a formal framework that is able to account for any type of mathematical expression. For the sake of a better understanding, we describe the approach in terms of online handwritten mathematical expressions such that primitives are strokes, but the generalization to offline expressions (printed or handwritten) is straightforward using connected components as primitives.

In this chapter, we first review previous proposals for mathematical expression recognition and introduce our model in Section 2.1. Then, Section 2.2 describes the statistical framework of the integrated grammar-based approach. This framework derives two different probabilities: the symbol likelihood that is described in Section 2.3, and the structural probability that is defined in Section 2.4.

Chapter Outline

2.1 Introduction

The problem of automatic mathematical expression recognition has been studied for decades [Anderson, 1967; Chou, 1989; Chan and Yeung, 2000; Zanibbi and Blostein, 2012]. Different approaches have been presented for this task. Some systems are based on recursively dividing the math formula into subexpressions by means of projection profiles [Okamoto and B., 1991], the X-Y cut algorithm [Ha et al., 1995] or using prior knowledge about the structure of math notation [Faure and Wang, 1990]. These approaches effectively decompose the mathematical expression into smaller subproblems. However, some cases like square roots require special treatment, and sloped expressions can be challenging for this methodology.

Another group of methods are based on trees or graphs. Eto and Suzuki [2001] developed a model for printed math expression recognition that computed the minimum spanning tree of a network representation of the expression. Tapia and Rojas [2004] presented a proposal for online recognition also based on constructing the minimum spanning tree and using symbol dominance. Zanibbi et al. [2002] recognized an expression as a tree, and proposed a system based on a sequence of tree transformations. Lehmberg et al. [1996] defined a net such that the sequence of symbols within the handwritten expression was represented by a path through the graph. Shi et al. [2007] presented a similar system where symbol segmentation and recognition were tackled simultaneously based on graphs. They then generated several symbol candidates for the best segmentation, and the recognized expression was computed in the final structural analysis [Shi and Soong, 2008]. This group of approaches generally results in efficient algorithms for recognizing formulas, and trees and graphs are proper models for representing mathematical expressions. However, context-free dependencies are not naturally modelled in most of these structures. Also, some approaches require a one-dimensional order, but math notation is 2D. Therefore, the order is often achieved by detecting baselines and exploiting the left-to-right reading order. But errors in the baseline detection cannot be solved in further steps. Another option to obtain a one-dimensional order in online recognition is to assume that symbols are written with consecutive strokes, which limits the set of accepted inputs.

Given the well-defined structure of math notation, many approaches are based on grammars because they constitute a natural way to model this problem. In fact, the first proposals on math expression recognition were grammar-based [Anderson, 1967; Chou, 1989]. Since then, different studies have been developed using different types of grammars. For instance, Chan and Yeung

[2001] used definite clause grammars, the model of Lavirotte and Pottier [1998] was based on graph grammars, Yamamoto et al. [2006] presented a system using Probabilistic Context-Free Grammars (PCFG), and MacLean and Labahn [2013] developed an approach using relational grammars and fuzzy sets. Despite previous approaches use different types of grammars, the methodology is based on modelling more complex structural relations by means of rules that combine problems creating larger hypotheses. In this thesis we will focus on models based on PCFG, because the structure of math notation can be naturally modelled using context-free grammars, and specifically PCFG because this well-known formalism will allow us to develop a formal statistical framework.

Proposals based on PCFG use grammars to model the structure of the expression, but the recognition systems are different. Garain and Chaudhuri [2004] proposed a system that combines online and offline information in the structural analysis. First, they created online hypotheses based on determining baselines in the input expression, and then offline hypotheses using recursive horizontal and vertical splits. Finally they used a context-free grammar to guide the process of merging the hypotheses. Yamamoto et al. [2006] presented a version of the CYK algorithm for parsing 2D-PCFGs with the restriction that symbols and relations must follow the writing order. They defined probability functions based on a region representation called "hidden writing area". Průša and Hlaváč [2007] described a system for offline recognition using 2D context-free grammars. Their proposal was penalty-based such that weights were associated with regions and syntactic rules. The model proposed by Awal et al. [2014] considers several segmentation hypotheses based on spatial information, and the symbol classifier has a rejection class in order to avoid incorrect segmentations. All these previous approaches were based on PCFG, but the resulting models have significant differences. Moreover, the lack of public datasets and standard metrics made that the performance of these proposals was not compared. Based on these studies, we wanted to develop our own approach for math expression recognition.

During the development of this thesis, we first tackled the problem of recognizing printed mathematical expressions. We developed an initial model based on connected components and 2D-PCFG parsing [Álvaro et al., 2011]. Afterwards, we moved on to online handwritten mathematical expression recognition with the emergence of the first CROHME competition [Mouchère et al., 2011], as it became an active research field. Our first approach was to apply the model we had developed for printed expressions to handwritten expressions. This way, by computing the connected components of the input strokes and

using specific classifiers and features for handwritten symbols, we obtained a parser for online handwritten math expressions [Álvaro et al., 2014]. However, this approach had problems regarding symbol segmentation and recognition and it did not take advantage of the online information.

That model tackled symbol segmentation by computing the connected components of the input strokes and merging them using productions of the grammar (e.g. an equals sign is a line below another line). However, this strategy required consideration of additional classes for symbol composition (e.g. an i without the dot or linked letters in functions) and finding proper spatial relations for their combination. Moreover, it could not account for touching symbols or symbols with segmentations that have not been taken into account with specific productions in the grammar (for instance, broken symbols like π in Fig. 1.2). Thus, segmentation was not a hidden variable but depended on design decisions of the grammar and symbol composition.

In order to overcome the problems of this segmentation methodology, we moved towards a model based on stroke primitives instead of connected components. This way we use the time information of the input and it represents a more powerful model to tackle symbol segmentation. But this strategy also introduces more challenges in order to determine which sets of strokes compose a mathematical symbol. We wanted to develop an approach based on parsing 2D-PCFG where symbol segmentation and recognition was integrated following an approach similar to [Shi et al., 2007; Luo et al., 2008]. The solution presented by Shi et al. [2007] combined several stochastic sources of information on symbol segmentation, symbol recognition and spatial relationships in order to determine the best overall segmentation and recognition. However, it had the restriction that symbols must be written with consecutive strokes in time and structural analysis was performed as a decoupled step. We pursue a formal model such that mathematical expressions can be written in any order.

As a result, in this thesis we describe an integrated approach based on parsing 2D-PCFG using strokes as primitives with no time order assumptions. Our proposal integrates several stochastic information sources in order to globally determine the most likely mathematical expression. In this advanced framework, segmentation becomes a hidden variable, and the statistical framework developed for online handwritten mathematical expressions can be applied to offline recognition by considering connected components as primitives instead of strokes, and pixels instead of points.

2.2 Statistical Framework

In online handwritten math expression recognition, the input is a sequence of strokes, and these strokes are in themselves a sequence points. Fig. 2.1 shows an example of the input for a mathematical expression. As you can see, the temporal sequence of strokes does not correspond to the sequence of symbols that it represents. For example, we can see that the user first wrote the subexpression $x - y$, then the user added the parentheses and its superscript $(x-y)^2$, finally converting the subtraction into an addition $(x+y)^2$. This example shows that some symbols might not be made up of consecutive strokes (e.g. the + symbol in Fig. 2.1). This means that the mathematical expression would not be correctly recognized if it was parsed monotonically with the input, i.e. processing the strokes in the order in which they were written. Meanwhile, the sequence of symbols that make up a subexpression does not have to respect the writing order (e.g. the parentheses and the subexpression they contain in Fig. 2.1).

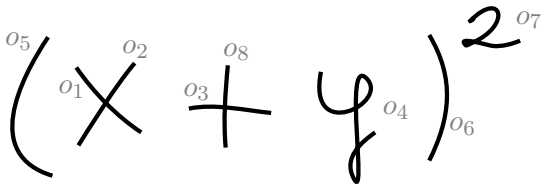


Figure 2.1: Example of input for an online handwritten mathematical expression. The order of the input sequence of strokes is labeled ($\boldsymbol{o} = o_1 o_2 \dots o_8$).

Given a sequence of input strokes, the output of a mathematical expression recognizer is usually a sequence of symbols [Shi et al., 2007]. However, we consider that a significant element of the output is the structure that defines the relationship between the symbols making up the final mathematical expression. As mentioned above, we propose modeling the structural relationships of a mathematical expression using a statistical grammatical model. By doing so, we define the problem of mathematical expression recognition as obtaining the most likely parse tree given a sequence of strokes. Fig. 2.2 shows a possible parse tree for the expression given in Fig. 2.1, where we can observe that a (context-free) structural model would be appropriate due to, for instance, structural dependencies in bracketed expressions. The output parse tree represents the structure that relates all the symbols and subexpressions

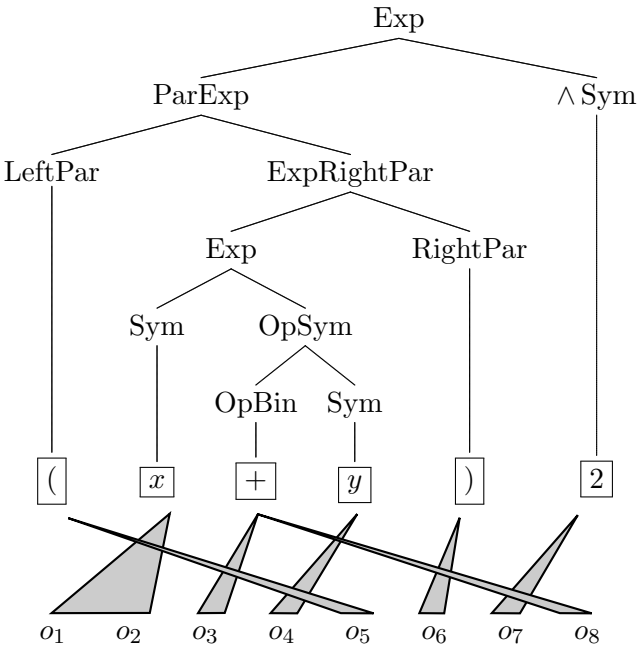


Figure 2.2: Parse tree of expression $(x + y)^2$ given the input sequence of strokes described in Fig. 2.1. The parse tree represents the structure of the math expression and it produces the 6 recognized symbols that account for the 8 input strokes.

that make up the input expression. The parse tree derivation produces the sequence of pre-terminals that represent the recognized mathematical symbols. Furthermore, to generate this sequence of pre-terminals, we must take into account all stroke combinations in order to form the possible math symbols.

Taking these considerations into account, two main problems have been observed. First, the segmentation and recognition of symbols is closely related to the alignment of mathematical symbols to strokes. Second, the structural analysis of a math expression addresses the problem of finding the parse tree that best accounts for the relationships between different mathematical symbols (pre-terminals). Obviously, these two problems are closely related. Symbol recognition is influenced by the structure of the mathematical expression, and detecting the structure of the math expression strongly depends on the segmentation and recognition of symbols. For these reasons we propose an integrated strategy that computes the most likely parse tree while simultaneously solving symbol segmentation, symbol recognition and the structural

analysis of the input.

Formally, let an online mathematical expression be a sequence of N strokes $\boldsymbol{o} = o_1 o_2 \dots o_N$. We state the mathematical expression recognition as a structural parsing problem such that the goal is to obtain the most likely parse tree t that accounts for the input sequence of strokes $\boldsymbol{o}$:

$$\hat{t} = \arg\max_{t \in \mathcal{T}} \; p(t \mid \boldsymbol{o})$$

where $\mathcal{T}$ represents the set of all possible parse trees.

At this point, we consider the sequence of mathematical symbols $\boldsymbol{s} \in \mathcal{S}$ as a hidden variable, where $\mathcal{S}$ is the set of all possible sequences of symbols (pre-terminals) produced by the parse tree t: $\boldsymbol{s} = \text{yield}(t)$. This can be formalized as follows:

$$\hat{t} = \arg\max_{t \in \mathcal{T}} \sum_{\substack{\boldsymbol{s} \in \mathcal{S} \\ \boldsymbol{s} = \text{yield}(t)}} p(t, \boldsymbol{s} \mid \boldsymbol{o})$$

If we approximate the probability by the maximum probability parse tree, and assume that the structural part of the equation depends only on the sequence of pre-terminals $\boldsymbol{s}$, the final target expression is

$$\hat{t} \approx \arg\max_{t \in \mathcal{T}} \max_{\substack{\boldsymbol{s} \in \mathcal{S} \\ \boldsymbol{s} = \text{yield}(t)}} p(\boldsymbol{s} \mid \boldsymbol{o}) \cdot p(t \mid \boldsymbol{s}) \tag{2.2.1}$$

such that $p(\boldsymbol{s}|\boldsymbol{o})$ represents the observation (symbol) likelihood and $p(t|\boldsymbol{s})$ represents the structural probability.

This problem could be solved in two steps. First, by calculating the segmentation of the input into mathematical symbols and, second, by computing the structure that relates all recognized symbols [Zanibbi et al., 2002]. However, in this study we propose an integrated strategy for computing Eq. (2.2.1) where symbol segmentation, symbol recognition and the structural analysis of the input expression are globally determined. This way, all the information is taken into account in order to obtain the most likely mathematical expression.

Next, in Section 2.3 we define the observation model that accounts for the probability of recognition and segmentation of symbols, $p(\boldsymbol{s}|\boldsymbol{o})$. The probability that accounts for the structure of the math expression $p(t|\boldsymbol{s})$ is described in Section 2.4. Finally, the algorithm that, for a given input $\boldsymbol{o}$, computes the mathematical expression $\hat{t}$ that maximizes the probability of Eq. 2.2.1 is presented in Chapter 5.

2.3 Symbol Likelihood

As we have seen, in the recognition of online handwritten math expressions, the input is a sequence of strokes $\boldsymbol{o} = o_1 o_2 \dots o_N$, which encodes a sequence of pre-terminals $\boldsymbol{s} = s_1 s_2 \dots s_K$, $(1 \leq K \leq N)$ that represents the mathematical symbols. A symbol is made up of one or more strokes. Some approaches assumed that users always write a symbol with consecutive strokes [Shi et al., 2007; Yamamoto et al., 2006]. Although this assumption may be true in many cases, it constitutes a severe constraint that means that these models cannot account for symbols composed of non-consecutive written strokes. For example, the plus sign (+) in the expression in Fig. 2.1 is made up of strokes o_3 and o_8 and would not be recognized by a model that incorporated this assumption.

In this section we define a symbol likelihood model that is not based on time information but rather spatial information. This model is therefore able to recognize math symbols made up of non-consecutive strokes. Given a sequence of strokes, testing all possible segmentations could be unfeasible given the high number of possible combinations. However, it is clear that only strokes that are close together will form a mathematical symbol, which is why we tackle the problem using the spatial and geometric information available since, by doing so, we can effectively reduce the number of symbol segmentations considered. The application of this intuitive idea is detailed in Section 3.2.2.

Before defining the segmentation strategy adopted for modeling the symbol likelihood, we must introduce some preliminary formal definitions.

Definition 2.3.1. Given a sequence of N input strokes $\boldsymbol{o}$, and the set containing them, $\text{set}(\boldsymbol{o}) = \{o_i \mid i : 1 \dots N\}$, a *segmentation* of $\boldsymbol{o}$ into K segments is a partition of the set of input strokes

$$b(\boldsymbol{o}, K) \;=\; \{\; b_i \mid i : 1 \dots K \;\}$$

where each b_i is a set of (possibly non-consecutive) strokes representing a segmentation hypothesis for a given symbol.

Definition 2.3.2. We define $\mathcal{B}_K$ as the set of all possible segmentations of the input strokes $\boldsymbol{o}$ in K parts. Similarly, we define the set of all segmentations $\mathcal{B}$ as:

$$\mathcal{B} = \bigcup_{1 \leq K \leq N} \mathcal{B}_K$$

Once we have defined the set of all possible segmentations $\mathcal{B}$, we want to calculate the probability of the segmentation and recognition of symbols

for a given sequence of strokes $\boldsymbol{o}$. In Eq. (2.2.1), we can define a generative model $p(\boldsymbol{s}, \boldsymbol{o})$, rather than $p(\boldsymbol{s}|\boldsymbol{o})$, because, given that the term $p(\boldsymbol{o})$ does not depend on the maximization variables $\boldsymbol{s}$ and t, we can drop it. The next step is to replace the sequence of N input strokes $\boldsymbol{o}$ by its previously defined set of segmentations, $b = b(\boldsymbol{o}, K) \in \mathcal{B}_K$ where $1 \leq K \leq N$. Finally, given K, we define a hidden variable that limits the number of strokes for each of the K pre-terminals (symbols) that make up the segmentation, $\boldsymbol{l} : l_1 \dots l_K$. Each l_i falls within the range $1 \leq l_i \leq \min(N, L_{\max})$, where $L_{\max}$ is a parameter that constrains the maximum number of strokes that a symbol can have.

$$p(\boldsymbol{s}, \boldsymbol{o}) = \sum_{1 \leq K \leq N} \sum_{b \in \mathcal{B}_K} \sum_{\boldsymbol{l}} p(\boldsymbol{s}, b, \boldsymbol{l})$$

In order to develop this expression, we factor it with respect to the number of pre-terminals (symbols) and assume the following constraints: 1) we approximate the summations by maximizations, 2) the probability of a possible segmentation depends only on the spatial constraints of the strokes it is made up of, 3) the probability of a symbol depends only on the set of strokes associated with it, and 4) the number of strokes for a pre-terminal depends only on the symbol it represents. Therefore, the probability becomes

$$p(\boldsymbol{s}, \boldsymbol{o}) \approx \max_{K} \max_{b \in \mathcal{B}_K} \max_{\boldsymbol{l}} \prod_{i=1}^{K} p(b_i)\, p(s_i \mid b_i)\, p(l_i \mid s_i) \tag{2.3.1}$$

From Eq. (2.3.1) we can conclude that we need to define three models: a *symbol segmentation model*, $p(b_i)$, a *symbol classification model*, $p(s_i|b_i)$, and a *symbol duration model*, $p(l_i|s_i)$. These three models for symbol recognition are discussed in depth in Chapter 3.

2.4 Structural Probability

The statistical framework described in this thesis defines the problem of recognizing a mathematical expression as finding the most likely parse tree t that accounts for the input strokes $\boldsymbol{o}$. Formally, the problem is stated in Eq. (2.2.1) such that two probabilities are required. In the previous section we presented the calculation of the *symbol likelihood* $p(\boldsymbol{s}|\boldsymbol{o})$. In this section we will define the *structural probability* $p(t|\boldsymbol{s})$.

Although the natural way to compute the most likely parse tree of an input sequence would be to define probabilistic *parsing models* $p(t|\boldsymbol{s})$, in the literature, this problem has usually been tackled using generative models $p(t, \boldsymbol{s})$

(*language models*) and, more precisely, *grammatical models* [Manning and Schütze, 1999]. Next we define a generative model $p(t, \boldsymbol{s})$ based on a two-dimensional extension of the well-known context-free grammatical models.

2.4.1 2D Probabilistic Context-Free Grammars

A context-free model is a powerful formalism able to represent the structure of natural languages. It is an appropriate model to account for math notation given the structural dependencies existing between different elements in an expression (for instance, the parentheses in Fig. 2.1). We will use a two-dimensional extension of Probabilistic Context-Free Grammars (2D-PCFG), a well-known formalism widely used for math expression recognition [Anderson, 1967; Chou, 1989; Yamamoto et al., 2006; Awal et al., 2014].

Definition 2.4.1. A *Context-Free Grammar* (CFG) G is a 4-tuple $(\mathcal{N}, \Sigma, S, \mathcal{P})$ where $\mathcal{N}$ is a finite set of nonterminal symbols, Σ is a finite set of terminal symbols ($\mathcal{N} \cap \Sigma = \emptyset$), $S \in \mathcal{N}$ is the start symbol of the grammar, and $\mathcal{P}$ is a finite set of rules: $A \to \alpha$, $A \in N$, $\alpha \in (\mathcal{N} \cup \Sigma)^+$.

A CFG in Chomsky Normal Form (CNF) is a CFG in which the rules are of the form $A \to BC$ or $A \to a$ (where $A, B, C \in \mathcal{N}$ and $a \in \Sigma$).

Definition 2.4.2. A *Probabilistic Context-Free Grammar* (PCFG) $\mathcal{G}$ is defined as a pair (G, p), where G is a CFG and $p : P \to]0, 1]$ is a probability function of rule application such that $\forall A \in \mathcal{N} : \sum_{i=1}^{n_A} p(A \to \alpha_i) = 1$, where n_A is the number of rules associated with nonterminal symbol A.

Definition 2.4.3. A *Two-Dimensional Probabilistic Context-Free Grammar* (2D-PCFG) is a generalization of a PCFG, where terminal and nonterminal symbols describe two-dimensional regions. This grammar in Chomsky Normal Form results in two types of rules: terminal rules and binary rules. First, the terminal rules $A \to a$ represent the mathematical symbols which are ultimately the terminal symbols of 2D-PCFG. Second, the binary rules $A \xrightarrow{r} BC$ have an additional parameter r that represents a given spatial relationship, and its interpretation is that regions B and C must be spatially arranged according to the spatial relationship r.

The application of this formalism to calculating the probability of the structure of a mathematical expression, defined as a parse tree in the statistical framework of this approach, is defined below.

2.4.2 Parse Tree Probability

The 2D-PCFG model allows us to calculate the structural probability of a mathematical expression in terms of the joint probability $p(t, \boldsymbol{s})$, such that in Chomsky Normal Form is computed as:

$$p(t, \boldsymbol{s}) = \prod_{(A \to a, t)} p(a \mid A) \prod_{(A \to BC, t)} p(BC \mid A)$$

where $p(\alpha|A)$ is the probability of the rule $A \to \alpha$ and represents the probability that α is derived from A. Moreover, $(A \to \alpha, t)$ denotes all rules $(A \to \alpha)$ contained in the parse tree t. In the defined 2D extension of PCFG, the composition of subproblems has an additional constraint according to a spatial relationship r. Let the spatial relationship r between two regions be a hidden variable. Then, the probability of a binary rule is written as:

$$p(BC \mid A) = \sum_{r} p(BC, r \mid A)$$

Therefore, by approximating summations by maximizations, and assuming that the probability of a spatial relationship depends only on the subproblems B and C involved, the structural probability of a mathematical expression becomes:

$$p(t, \boldsymbol{s}) \approx \prod_{(A \to a, t)} p(a \mid A) \tag{2.4.1}$$

$$\prod_{(A \to BC, t)} \max_{r} p(BC \mid A)\, p(r \mid BC) \tag{2.4.2}$$

where $p(a|A)$ and $p(BC|A)$ are the probabilities of the rules of the grammar, and $p(r|BC)$ is the probability that regions encoded by nonterminals B and C are arranged according to spatial relationship r. The estimation of the probabilities of the grammar and the definition of the spatial relationships probability are thoroughly detailed in Chapter 4.

Chapter

3 Symbol Recognition

The problem of recognizing mathematical expressions requires dealing with several tasks. Symbol classification is crucial in order to achieve good recognition results because, in the end, a mathematical expression is a set of symbols and their structural relationships. The symbol likelihood probability defined in the statistical framework for mathematical expression recognition resulted in three probability distributions: a symbol segmentation model, a symbol classification model and a symbol duration model. In this chapter we address the definition, estimation and evaluation of these models for both handwritten and printed mathematical symbols.

Defining the models for symbol recognition can be generalized if we formulate them in terms of primitives: strokes for online recognition and connected components for offline recognition. Therefore, the symbol duration model and the symbol segmentation model are formally defined for any type of mathematical expression. Symbol duration model is defined in Section 3.1, and the segmentation model as well as the considered admissible hypotheses are described in Section 3.2.

Regarding symbol classification, handwritten mathematical symbols can be either online or offline. In Section 3.3 we describe several sets of features for handwritten symbol classification, including online, offline and a combination of them. On the other hand, printed mathematical symbols present significant differences with respect to handwritten symbols. Thus, we detail specific features and classifiers for printed symbol recognition in Section 3.4.

Chapter Outline

3.1 Symbol Duration

An intuitive idea about symbol composition is that a given mathematical symbol is commonly represented by a certain number of primitives. For example, in online representation, a number 2 is usually a single-stroke symbol, while symbols x and $=$ are likely to be composed of two strokes, and symbol π is normally made up of three strokes. Likewise, in offline representation, a number 2 or a symbol x are usually composed of one connected component, while an equals sign $=$ is normally made up of two connected components, or a trigonometric function is likely to be composed of three (cos) or more connected components (sin).

The symbol duration model accounts for the probability that a mathematical symbol class s_i is usually made up of a certain number of primitives l_i. This is the probability $p(l_i|s_i)$ required in the symbol likelihood calculation of Eq.(2.3.1). Shi et al. [2007] proposed a simple way to calculate this probability from a set of labeled data as

$$p(l_i \mid s_i) = \frac{c(s_i, l_i)}{c(s_i)} \tag{3.1.1}$$

where $c(s_i, l_i)$ is the number of times the symbol s_i was composed of l_i primitives and $c(s_i)$ is the total number of samples of class s_i in the dataset used for estimation. We smoothed these probabilities in order to account for unseen events using add-one smoothing [Jurafsky and Martin, 2008].

3.2 Symbol Segmentation

One of the main challenges in mathematical expression recognition is determining the segmentation of the input into symbols. As presented in Section 1.2.1, formulae contain several symbols that can be encoded by one or more than one primitives. Although context is important for proper math symbol segmentation, we can decide whether or not a set of primitives is likely to represent a mathematical symbol based on, for example, their geometric and spatial information.

In this section the goal is to define a statistical model that provides the probability that a set of primitives can form a mathematical symbol. First, we introduce and review previous work in symbol segmentation. Second, we detail the segmentation model defined for mathematical symbols.

3.2.1 Introduction

In the literature, several segmentation strategies have been proposed in both handwritten and printed mathematical expression recognition. As discussed in Section 1.2.1, there are different problems concerning segmentation of mathematical symbols: noise, touching characters and symbols composed of more than one primitive.

Noise reduction in offline expressions can be tackled using image treatment techniques like median filters or morphological operations (dilation/erosion). These methods try to remove small connected components caused by image degradation while preserving the pixels that represent the foreground information. It can effectively reduce noise, but can also produce cases of touching characters that were not in the original image. In online recognition, the user can introduce noisy small strokes while handwriting an expression, that can also be preprocessed. In both online and offline recognition, math notation contains many small symbols (dots and diacritical marks) which can be difficult to distinguish from noise.

The segmentation of touching symbols in offline recognition is challenging and usually requires specific treatment. There are some proposals for tackling this problem. Nomura et al. [2003] used specific geometric features for detecting touching characters and the deviation from standard feature values in symbol classification. Garain and Chaudhuri [2005b] developed a predictive algorithm based on multifactorial analysis that selects possible cut-positions for segmenting touching symbols. Tian and Zhang [2007] detected and segmented touching math symbols using a method based on contour features and the ratio between width and height. A similar problem in online recognition is when two or more symbols are written with a single stroke (continuous handwriting), although this is uncommon in math expression recognition. Only letters in functions tend to share strokes (cos, lim, etc.), which is usually solved by considering functions as math symbol classes. In this thesis, we leave the segmentation of touching symbols as future work. A possible direction to deal with this issue in the current framework would be to work with oversegmented expressions.

After dealing with noise and touching characters, ideally, we would have clean inputs of mathematical expressions where each symbol is composed of one or more primitives. In the literature, some approaches for recognizing mathematical expressions are not directly based on analysis of primitives. Chou [1989] method had two stages. The *lexical access* stage performs the pattern matching between input regions of the image and a set of input templates using the Hamming distance. The parsing stage computes the recog-

nized expression according to a 2D stochastic context-free grammar. Other approaches consist in recursively dividing the input image. Faure and Wang [1990] divided the expressions according to the input data and the prior knowledge about the structure of math notation. Okamoto and B. [1991] applied the projection profile cutting algorithm to recursively split the input along both vertical and horizontal directions. Ha et al. [1995] proposed a segmentation method in two steps. First, the expression is processed top-down using the recursive X-Y cut algorithm. Second, a bottom-up process considering spatial relations between neighboring objects.

Regarding offline recognition methods based on connected components, Twaakyondo and Okamoto [1995] proposed an approach consisting in three main parts. First, symbol recognition was performed by labelling and merging connected components. Second, a specific structure processing step where certain type of expressions are analyzed. Finally, the fundamental structure processing step in which the global structure of the recognized mathematical expression is evaluated. Suzuki et al. [2003] developed a method such that symbol segmentation is first determined using a coarse-to-fine strategy with geometric features. A clustering technique where each connected component is appended to its nearest cluster is used to correct the initial segmentation. Finally, a network representing multiple hypotheses is constructed and the minimum spanning tree in that network is the recognized mathematical expression. Průša and Hlaváč [2007] presented a system based on parsing 2D grammars. The segmentation method consisted in computing several hypothesis based on connected components and a scanning window. Afterwards, the parsing process computed the expression according to the grammar minimizing a global penalty.

Methods based on recursively dividing the math expressions into subexpressions [Faure and Wang, 1990] are generally more suitable for printed expressions than for handwritten expressions. In online recognition, there are several approaches based on a graph representation that encodes several segmentation and recognition hypotheses, similar to the proposal for offline recognition by Suzuki et al. [2003]. Lehmberg et al. [1996] and Shi et al. [2007] defined a net or symbol graph such that the sequence of symbols within the handwritten expression was represented by a path through the graph. Both proposals assumed that symbols were composed of consecutive strokes in time. Matsakis [1999] created symbol segmentation hypotheses by computing the minimum spanning tree of the input strokes. Kosmala and Rigoll [1998] proposed solving the segmentation and recognition of mathematical symbols using the global decoding of Hidden Markov Models (HMM). Moreover, as several

approaches have used grammars for recognizing math notation, these proposals have segmentation strategies driven by grammars [Yamamoto et al., 2006; Průša and Hlaváč, 2007; Hu et al., 2012; MacLean and Labahn, 2013; Awal et al., 2014].

Generally, despite the great number of proposals, in most of them it is common to have a model that measures whether or not a hypothesis is likely to form a mathematical symbol. On the one hand, decoupled approaches compute symbol segmentation prior to performing the analysis of the expression, and errors in segmentation cannot be recovered in further steps. This is the case of methods that recursively divide the expressions or that perform the structural analysis after symbol segmentation is determined.

On the other hand, integrated approaches incorporate a probability distribution or a scoring function in the computation of the recognized formula, such that segmentation is determined globally. These probabilities or scores are commonly based on computing geometric features [Lehmberg et al., 1996; Shi et al., 2007; Toyozumi et al., 2004; MacLean and Labahn, 2013] or shape descriptors [Hu and Zanibbi, 2013] over primitives that can be combined to form the mathematical symbols.

In the statistical framework of the approach defined in this thesis, the segmentation is based on primitives and globally determined by the probability of the most likely tree. For the calculation of this probability we require a segmentation model. Below we define the set of admissible segmentation hypotheses and the segmentation model that accounts for the prior probability that a set of primitives can form a mathematical symbol.

3.2.2 Segmentation Model

The statistical framework developed in thesis defines an input mathematical expression as a sequence of observations $\boldsymbol{o} = o_1 o_2 \dots o_N$. Commonly this represents a sequence of input strokes in online recognition such that, as we have already discussed, the approach proposed in this thesis is not based on temporal information, but rather on spatial and geometric information. Consequently we can also apply this framework to offline recognition by enumerating the connected components in any order.

In Section 2.3 we defined the set of all possible segmentations $\mathcal{B}$. Given this definition of $\mathcal{B}$, we can see that its size is exponential on the number of primitives N. In this section we first explain how to effectively reduce the number of segmentations considered. Then, we describe the segmentation model used for computing the probability of a certain hypothesis $p(b_i)$ required in the calculation of the symbol likelihood of Eq. (2.3.1).

Taking into account all possible segmentations $\mathcal{B}$ for a given math expression could be unfeasible. In order to reduce this set, we use two concepts based on geometric and spatial information: *visibility* and *closeness*. Let us first introduce some definitions.

Definition 3.2.1. The distance between two primitives o_i and o_j can be defined as the Euclidean distance between their closest points (or pixels).

$$\mathrm{dist}(o_i, o_j) = \min_{p \in o_i, q \in o_j} \sqrt{(q_{.x} - p_{.x})^2 + (q_{.y} - p_{.y})^2}$$

Definition 3.2.2. A primitive o_i is considered *visible* from o_j if the straight line between the closest points (or pixels) of both primitives does not cross any other primitive o_k.

If a primitive o_i is not *visible* from o_j we consider that their distance is infinite. For example, given the expression in Fig. 2.1, the strokes visible from o_4 are o_3, o_6 and o_8.

Furthermore, when multiple primitives form a math symbol is because those primitives are spatially close. For this reason, we only consider segmentation hypotheses b_i where primitives are close to each other.

Definition 3.2.3. A primitive o_i is considered *close* to another primitive o_j if their distance is shorter than a given threshold b_{th}.

Using these definitions, we can characterize the set of admissible segmentation hypotheses.

Definition 3.2.4. Let G be an undirected graph such that each primitive is a node and edges only connect primitives that are *visible* and *close*. Then, a segmentation hypothesis b_i is admissible if the primitives it contains form a connected subgraph in G.

Consequently, a segmentation $\boldsymbol{b}(\boldsymbol{o}, K) = b_1 b_2 \dots b_K$ is admissible if each b_i is, in turn, admissible. These two geometric and spatial restrictions significantly reduce the number of possible symbol segmentations.

In addition to reducing the set of possible segmentations, we need a segmentation model in order to calculate the probability that a given set of primitives (segmentation hypothesis, b_i) forms a mathematical symbol. We define a segmentation model very similar to the concept of *grouping likelihood* proposed by Shi et al. [2007], extracting a set of geometric features for a given segmentation hypothesis b_i. First, for each primitive o_j of b_i, we compute the mean horizontal position, the mean vertical position and its size (calculated

as the maximum value of horizontal and vertical size). Then, for every pair of primitives in b_i we compute the difference between their horizontal positions, vertical positions, sizes, and their distance. The feature vector used for the segmentation model is obtained averaging these four values for each pair: *average horizontal distance* (d) *average vertical offset* (σ), *average size difference* (δ), and *average distance* (θ).

Shi et al. [2007] used a scoring function such that these features were normalized using a fixed threshold value, but this normalization made the features dependent on the resolution of the input data. In order to ensure that the features are resolution-independent, we normalized them by the diagonal of the *normalized symbol size* (defined in Section 5.2).

Once we have defined a set of normalized geometric features, the last step is computing the segmentation probability $p(b_i)$. Instead of the scoring function proposed by Shi et al. [2007], we trained a Gaussian Mixture Model (GMM) using positive samples $c = 1$ (the primitives of b_i can form a math symbol) and a GMM using negative samples $c = 0$ (the primitives of b_i cannot form a math symbol) from the set of all admissible segmentations $\mathcal{B}$. A segmentation hypothesis b_i is represented by the 4-dimensional normalized feature vector $\mathrm{g}(b_i) = [d, \sigma, \delta, \theta]$, and the probability $p(b_i)$ that a hypothesis b_i forms a math symbol is obtained as

$$p(b_i) = p_{\mathrm{GMM}}(c = 1 \mid \mathrm{g}(b_i)) \tag{3.2.1}$$

We selected GMM because they are a well-defined statistical model, fast to compute for classification, and our intuition about the defined geometric features is that they follow a normal distribution. Other statistical classifiers can be used in order to obtain this probability (e.g. support vector machines) where discriminative training could lead to better performance.

3.3 Handwritten Symbol Classification

In mathematical expression recognition, symbol classification is a crucial step. In this section we first introduce the problem and review related work on handwritten mathematical symbol recognition. Then, we detail features for online classification and several features for offline classification. Finally, two main classifiers are presented, and the evaluation of the different proposals is reported in Section 7.2.

3.3.1 Introduction

Handwritten mathematical symbol classification is a classical pattern recognition problem. A given sample (b_i) is first preprocessed, then several features $\mathcal{F}(b_i)$ are extracted and a certain classifier (θ) is used to compute the most likely symbol class (s) according to labeled data. The probability of the symbol classification model for a given class is obtained as the posterior probability provided by a classifier

$$p(s \mid b_i) = p_\theta(s \mid \mathcal{F}(b_i)) \tag{3.3.1}$$

A great number of classifiers have been proposed in the past for online mathematical symbols [Zanibbi and Blostein, 2012]. Thammano and Rugkunchon [2006] used artificial neural networks for online symbol classification using first an online stage to classify in four different groups, and then an offline stage for final recognition. Winkler [1996] combined three HMMs for handwritten symbol recognition. Given an online sample, three different sets of features (one online and two offline) were extracted and the HMM emission probabilities were later combined to produce the final classification. A similar approach was proposed by Keshari and Watt [2007] using Support Vector Machines (SVM) for this task. They produced the image representation of each sample and trained two classifiers: SVM for online classification and SVM for offline classification. The final recognition was obtained by combining both SVMs using a weight parameter. Garain and Chaudhuri [2004] extracted directional features from the online symbols and recognized them with two classifiers: Nearest Neighbor and HMM. Simistira et al. [2008] used a classifier based on template elastic matching distance, such that the math symbols were encoded using pen-direction features using the 8-level Freeman chain coding scheme. Luo et al. [2008] presented an advanced method for classification that considers the whole mathematical expression to build a graph representing several symbol segmentation and recognition hypothesis. The symbol classifier was a GMM per symbol class using histogram of oriented gradients as features.

Despite the great number of papers published, they were not directly comparable because they used private datasets and different experimentation. Fortunately, the publication of the MathBrush database [MacLean et al., 2011] and the CROHME competitions [Mouchère et al., 2011, 2012, 2013, 2014] provided good resources for handwritten mathematical symbol recognition and later approaches reported results using them. MacLean and Labahn [2010] developed an efficient elastic matching algorithm. Hu and Zanibbi [2011] used an online HMM classifier proposing a novel initialization method and a new feature. Davila et al. [2014] defined a wide set of features for online handwritten symbols: global geometric features, crossing features, and fuzzy histograms of points and orientations. The authors also compared the performance of AdaBoost, Random Forests and SVM classifiers. Julca-Aguilar et al. [2014] proposed fuzzy shape context and online features for classification with Multi Layer Perceptron neural networks. They also considered the rejection of false segmentation hypotheses.

The different methods presented for recognition can be grouped in two sets. First, global classification such that a set of features describes each symbol hypothesis and different classifiers can be used (e.g. nearest neighbor, SVM or ANN). Second, sequential classification such that symbol hypotheses are encoded as a series of feature vectors and sequential classifiers are required (e.g. HMMs or Recurrent Neural Networks). In the following sections we describe several sets of features for sequential classification of handwritten mathematical symbols, as well as two different classifiers. All these proposals are evaluated and compared to other approaches in Section 7.2.

3.3.2 Online Features

An online mathematical symbol given as a set of strokes b_i represents a sequence of points p in space. We extract online features as

$$\mathcal{F}(b_i) = f(p_1)f(p_2)\dots f(p_M); \qquad M = \sum_{o_j \in b_i} |o_j|$$

such that we concatenate the sequences of points of the strokes $o_j \in b_i$ according to the input order. Here we present two sets of features. First, a set of online features. Second, a set of hybrid features that extends the online features with offline information.

Online Features

For each point $p = (x, y)$, we compute seven time-based features proposed by Toselli et al. [2007] for handwritten text recognition:

- Normalized coordinates: (x, y) normalized values such that $y \in [0, 100]$ and the aspect-ratio of the sample is preserved.
- Normalized first derivatives: (x', y').
- Normalized second derivatives: (x'', y'').
- Curvature: k, the inverse of the radius of the curve in each point.

Finally, for each point p we obtain a feature vector $f_{\text{on}}(p)$ as:

$$f_{\text{on}}(p) = [\, x, y, x', y', x'', y'', k \,]$$

It should be noted that no resampling is required prior to the feature extraction process because first derivatives implicitly perform writing speed normalization [Toselli et al., 2007].

Hybrid Features

In order to complement the online features with offline information, we propose to extend them taking into account a context window in the offline representation centered in the considered point p (Figure 3.1). Given a context window of size $w \times w$ we projected the w^2 gray-scale values to d dimensions by using Principal Component Analysis (PCA). Thus, we obtained a set of hybrid features for online classification such that each point $p = (x, y)$ is represented by the following feature vector

$$f_{\text{hyb}}(p) = [\underbrace{x, y, x', y', x'', y'', k}_{\text{online}}, \underbrace{v_1, v_2, \ldots, v_d}_{\text{offline}}]$$

The image representation of an online sample was rendered setting the height of the image to a fixed value h and preserving the aspect ratio (up to $5h$ to prevent symbols like a fraction bar from creating too wide images). Then, the image was produced through linear interpolation among every consecutive points within each stroke. Finally, a 3×3 Gaussian filter window was applied to slightly smooth the image, producing the final gray-scale image (see Figure 3.1) such that each pixel is in the range $[0, 255]$.

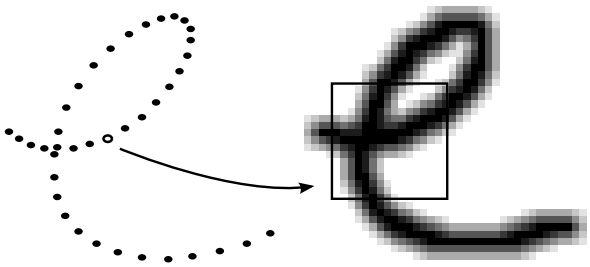


Figure 3.1: The online features are extended with offline information using a context-window centered on each point in the rendered image.

3.3.3 Offline Features

In this section we describe several sets of offline features for handwritten mathematical symbol classification. These features are extracted from samples encoded as images, but offline features are also useful in order to complement online information since we can easily produce the image representing an online mathematical symbol. We refer to these features by the name of the group or university that has developed them: PRHLT, FKI and RWTH.

A mathematical symbol represented as a set of primitives b_i, either connected components or strokes, can be encoded as an image $\mathcal{I}$ of $W \times H$ pixels. Given this image, we extract offline features for each column $\mathcal{I}_j$ as

$$\mathcal{F}(b_i) = f(\mathcal{I}_1)f(\mathcal{I}_2)\dots f(\mathcal{I}_W)$$

PRHLT Offline Features

In this section we describe the offline features that the Pattern Recognition and Human Language Technologies (PRHLT) group has used for many handwritten text recognition tasks [Toselli et al., 2004a].

Given the image of a math symbol (Fig. 3.2.a) it is transformed into a sequence of feature vectors as follows. First, the image is divided into a grid of small square cells sized a small fraction of the image height ($H/20$). Then for each cell we compute three different values (Fig. 3.2): normalized gray level (b), horizontal gray-level derivative (c) and vertical gray-level derivative (d). The feature extraction was extended to a 5×5 window centered at the current cell to obtain smoothed values. The values are weighted by a two-dimensional Gaussian function in (b) and a one-dimensional Gaussian

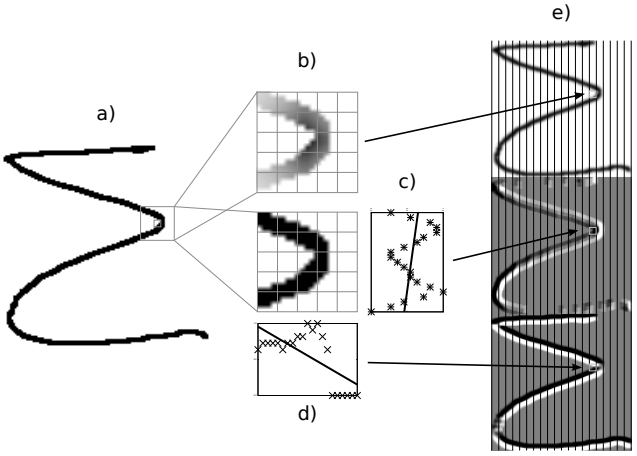


Figure 3.2: Example of PRHLT offline features computation. Given an image of a math symbol (a) we compute the normalized gray level (b), horizontal gray-level derivative (c) and vertical gray-level derivative (d). Each column of the stacked images (e) represents a feature vector.

function in (c) and (d). The derivatives are computed by least squares fitting a linear function. Finally, we stack the three different images that result after computing these values for each cell (see Fig. 3.2.e). Thus, each column from left to right represents a feature vector f_{prhlt} of 60 dimensions.

FKI Offline Features

The FKI features proposed by Marti and Bunke [2001] are a well-known set of geometric features that has been used for years in handwriting recognition. Given a binary image of height H, we compute 9 geometrical values c_i for each column x as:

- Number of black pixels in the column:

$$c_1(x) = \sum_{y=1}^{H} \mathcal{I}(x, y)$$

- Center of gravity of the column:

$$c_2(x) = \frac{1}{H} \sum_{y=1}^{H} y \cdot \mathcal{I}(x, y)$$

- Second order moment of the column:

$$c_3(x) = \frac{1}{H^2} \sum_{y=1}^{H} y^2 \cdot \mathcal{I}(x, y)$$

- Position of the upper contour in the column:

$$c_4(x) = \min\{y \mid \mathcal{I}(x, y) = \text{black}\}$$

- Position of the lower contour in the column:

$$c_5(x) = \max\{y \mid \mathcal{I}(x, y) = \text{black}\}$$

- Orientation of the upper contour in the column:

$$c_6(x) = \frac{c_4(x+1) - c_4(x-1)}{2}$$

- Orientation of the lower contour in the column:

$$c_7(x) = \frac{c_5(x+1) - c_5(x-1)}{2}$$

- Number of black-white transitions in the column:

$$c_8(x) = \text{NT}_{\text{black}\rightarrow\text{white}}\, \mathcal{I}(x, y);\ 1 \leq y \leq H$$

- Number of black pixels between the upper and lower contours:

$$c_9(x) = \sum_{c_4(x)<y<c_5(x)} \mathcal{I}(x, y)$$

Finally, given a binary image of size $W \times H$, for each column x such that $1 \leq x \leq W$ this definition will generate a 9-dimensional feature vector as:

$$f_{\text{fki}}(\mathcal{I}_x) = [c_1(x), c_2(x), \ldots, c_9(x)]$$

RWTH Offline Features

The RWTH features [Dreuw et al., 2009] have been extensively used in handwriting recognition. Given an image of height H, this set of features is simply computed using a sliding window of width w from each column left to right (see Fig. 3.4a). Then, the $w \times H$ values of each extracted window are projected to D dimensions by using PCA. Finally each projected window represents a feature vector f_{rwth}.

Lately, the vertical repositioning method [Giménez et al., 2010] has provided very good results in handwriting recognition [Doetsch et al., 2012], therefore we decided to also test this method. We first computed the vertical center of gravity of each extracted window. Then, we repositioned the window such that its vertical center is aligned with the computed center of gravity. Fig. 3.4 shows an example of the RWTH features computed with and without vertical repositioning.

Polar Offline Features

Shape descriptors are a well-known representation that have been used for multiple applications [Yang et al., 2008]. Su et al. [2013] proposed a feature extraction based on polar histograms for handwritten text recognition. We defined similar features but we use circles instead of ellipses and equally spaced distances instead of log-distances, as a simpler first application of these descriptors to math symbol classification.

The polar descriptor centered in a point p was defined as follows. We draw n circles with radii equally spaced up to the maximum radius r. Moving counterclockwise, draw radii dividing each circle into m equal arcs. This descriptor is encoded as a matrix such that each row represents a circle and each column represent the angle starting from zero degrees. Figure 3.3 shows an example of polar descriptor.

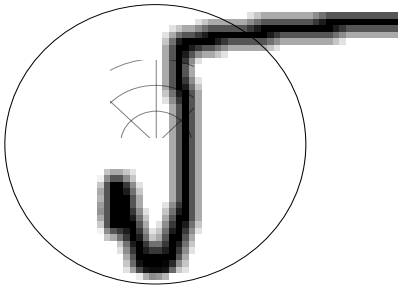


Figure 3.3: Example of polar descriptor with $n = 5$ circles and $m = 8$ arcs in a particular column of a square root symbol.

The idea is to use a sliding polar descriptor instead of the sliding window used in the previously described RWTH features. Given a descriptor, each of the $n \times m$ bins contains the number of foreground pixels that fall into that bin normalized by the total number of pixels in the descriptor. Thus, each feature vector f_{polar} is computed as the projection of that polar histogram to

D dimensions using PCA. Given an image of size $W \times H$, the feature vectors are computed from each column left to right setting the center of the polar descriptor on the center of each column, i.e. at row $H/2$.

We also wanted to apply the vertical repositioning method to these features. Given a column x and radius r, the standard polar features are centered in $p = (x, H/2)$. Following the methodology of the RWTH features, we computed the vertical centroid y' of the window from column $(x - r)$ to $(x + r)$ and then the descriptor center is set to $p = (x, y')$. Fig. 3.5 shows an example of the polar histograms computed with and without vertical repositioning.

3.3.4 Classifiers

In the introduction of this section we reviewed that several classifiers have been used for handwritten mathematical symbol recognition. In this thesis we test two different sequential classifiers for this task: Hidden Markov Models and Recurrent Neural Networks. The evaluation of these classifiers for handwritten symbol classification is reported in Section 7.2.

Hidden Markov Models

Hidden Markov Models (HMMs) are a statistical model such that if we use them to predict the next observation in a sequence, the distribution of predictions will depend only on the value of the immediately preceding observation and will be independent of all earlier observations [Bishop, 2006]. If we have a set of observations, we can estimate the parameters of an HMM using maximum likelihood by means of the Baum-Welch algorithm [Baum, 1972].

HMMs have extensively been used in speech recognition [Jurafsky and Martin, 2008], handwriting recognition [Plamondon and Srihari, 2000] and natural language processing [Manning and Schütze, 1999]. In this thesis, we use left-to-right HMMs for mathematical symbol classification, thus observation sequences are assumed to be symbols and no segmentation and language models are required. We can efficiently compute the posterior probability for symbol s of a given sample b_i with the well-known *forward-backward* algorithm [Wessel et al., 2001]

$$p(s \mid b_i) = p_{\text{HMM}}(s \mid \mathcal{F}(b_i))$$

Figure 3.4: Example of the RWTH offline features for a square root symbol using a sliding window of $w = 11$ pixels. Each window represents a feature vector after projecting it to D dimensions with PCA.

Figure 3.5: Example of polar features for a square root symbol using $n = 5$ circles and $m = 12$ arcs. The gray-scale colors of the bins in the descriptors represent the values of the polar histogram, from zero (white) to the maximum value in that descriptor (black).

Recurrent Neural Networks

Sequential classification has traditionally been tackled using HMMs. However, in the last years RNNs have brought more attention to the research community. RNNs are a connectionist model containing a self-connected hidden layer. The recurrent connection provides information of previous inputs such that the network can benefit from past contextual information [Pearlmutter, 1989]. The Long Short-Term Memory (LSTM) advanced RNN architecture allows that cells can access to context information over long periods of time. This is achieved by using a hidden layer composed of recurrently connected subnets, called memory blocks [Graves et al., 2009].

Bidirectional RNNs [Schuster and Paliwal, 1997] have two separate hidden layers that allow the network to access context information in both time directions: one hidden layer processes the input sequence forward and the other processes it backward. The combination of bidirectional RNNs and the LSTM architecture results in BLSTM-RNNs. These networks have outperformed standard RNNs and HMMs in handwriting text recognition [Graves et al., 2009] and also they are faster than HMMs in terms of classification speed.

In this thesis we used BLSTM-RNN for mathematical symbol classification. The RNN was trained in a frame-based approach, i.e. given an input sequence $\mathcal{F}(b_i)$ composed of n vectors such that $\mathcal{F}(b_i) = f_1 f_2 \dots f_n$, the network computes for each class s the posterior probability $p(\, s \mid f_i \,)$, where f_i represents a feature vector. Then, we obtained the posterior probability per symbol after the following normalization:

$$p(s \mid b_i) = p_{\text{RNN}}(s \mid \mathcal{F}(b_i)) = \frac{1}{n} \sum_{i=1}^{n} p(\, s \mid f_i \,) \tag{3.3.2}$$

where the probability per symbol for class c is computed as its average probability per frame.

3.4 Printed Symbol Classification

Automatic recognition of printed mathematical symbols is a fundamental problem for recognizing mathematical expressions. In the following sections we present the related work on this field and describe different features and classifiers for this task. The methods proposed for printed math symbol classification are evaluated in Section 7.3.

3.4.1 Introduction

Recognition of typeset mathematical symbols is a difficult problem due to there is a large number of symbols, with different font-types (e.g. roman, italic, calligraphic) and different sizes within a mathematical expression. As defined by Eq. (3.3.1) for handwritten symbols, the probability of the symbol classification model is computed as the posterior probability of a given classifier. Since there are significant differences between printed and handwritten symbols, specific features and classifiers for both problems are convenient. Especially, printed symbols are more regular than handwritten symbols.

Several techniques have been proposed for the offline recognition of printed mathematical symbols. Okamoto and Miyazawa [1992] used a simple template matching technique for classifying mathematical symbols. Lee and Lee [1993] proposed a classification in two stages. First, a coarse classification algorithm was used to reduce the number of candidates. Second, the character with the highest similarity is selected using 13 features. Fateman et al. [1996] represented each math symbol by a vector of 27 features and classified them using minimum Euclidean distance. They divided the bounding box of the symbols into a 5×5 grid, and then counted the percentage of black pixels in each cell. The additional two features are the height-to-width ratio and the absolute height in pixels of the bounding box. Garain et al. [2004] developed a method that first looked for the presence of certain simple primitives to recognize symbols by combining several disconnected primitives. The symbols that cannot be identified by the former classifier are passed to a combination of three classifiers: a run-number based classifier, a grid-based classifier, and a wavelet decomposition based classifier. Malon et al. [2008] presented both directional features and density features for classification with SVM using different kernels.

Many classification methods have been proposed in the literature. Nevertheless, there are very few studies that compare different classification techniques on the same database and under the same experimental conditions. Below we describe the feature extraction process used to train several classi-

fiers for printed math symbol classification. The performance of the proposed classifiers will be assessed in Section 7.3 on two public databases.

3.4.2 Feature Extraction

Two types of classifiers will be used for symbol classification. On the one hand, models that classify samples represented by a feature vector of fixed size. On the other hand, models for sequence classification.

Given a region of a gray-scale image encoding a printed mathematical symbol, we simply normalize it to 15×15 pixels. As a result, we obtain a 225-dimensional feature vector for classification such that each value is in the range $[0, 255]$. This simple representation is appropriate because printed mathematical symbols are quite regular, thus, little variations are expected for different samples of the same symbol.

Regarding sequence classification, we applied the feature extraction process used in handwritten text recognition by Toselli et al. [2004a] that we have also been applied for handwritten math symbol classification (see Section 3.3.3). Given the $W \times H$ image of a mathematical symbol, it is divided into a grid of small square cells sized a small fraction of the image height ($H/20$). Each cell is characterized by the following values (Fig. 3.6): normalized gray level (b), horizontal gray-level derivative (c) and vertical gray-level derivative (d). To obtain smoothed values of these features, the extraction process is extended to a 5×5-cell window centered at the current cell and weighted by a two-dimensional Gaussian function in (b) and a one-dimensional Gaussian function in (c) and (d). The derivatives are computed by least squares fitting a linear function. Columns of cells are processed from left to right such that a 60-dimensional feature vector is obtained from each column by stacking the values computed in its constituent cells. Figure 3.6e shows a graphical representation of the computed features.

3.4.3 Classifiers

In this section we present four classifiers for printed math symbol classification. We compare classical techniques with other techniques that have not been explored for this task. Below we briefly describe the proposed classifiers used for mathematical symbol recognition.

Nearest Neighbor

The k-Nearest-Neighbor (k-NN) rule is a very popular pattern classifier that provides good results when the number of prototypes is large. Each class is represented by several samples, such that a new sample is classified into the class obtained by majority vote of its k nearest neighbor prototypes. In this task we used the Euclidean distance to determine the nearest sample in the training set.

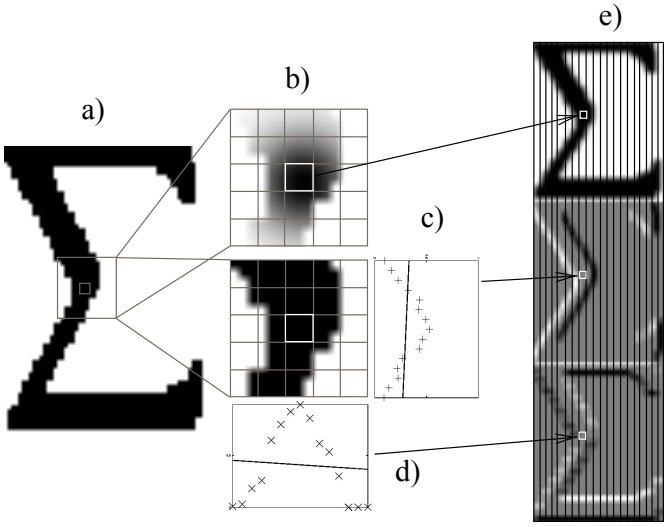


Figure 3.6: Example of feature extraction for sequence classification.

Weighted Nearest neighbor

The Weighted Nearest Neighbor (WNN) classifier was proposed by Paredes and Vidal [2006] as an improvement of the classical 1-NN. A discriminative technique is used to learn a weighted distance by using the 1-NN rule with a training set. A distance weighting scheme is proposed which can independently emphasize prototypes and/or features. Several alternatives are considered in [Paredes and Vidal, 2006]: using a different weight for each prototype, using a different weight for each class and feature, or using a combination of the previous alternatives. In this task, there are training samples more representative than others, and also in symbol representation the importance of each pixel is different. Consequently, it is reasonable to weight both the

prototypes and the features for each class.

The experimental results that are reported in Paredes and Vidal [2006] with the UCI and Statlog corpora were comparable to or better than those obtained by other state-of-the-art classification methods.

Support Vector Machines

Support Vector Machine (SVM) is a maximum margin classifier that has demonstrated to be a powerful formalism for many recognition tasks. SVM became popular for solving problems in classification, regression, and novelty detection. An important property of SVMs is that the determination of the model parameters corresponds to a convex optimization problem, and so any local solution is also a global optimum [Bishop, 2006].

In this work we used the multi-class SVM described in [Crammer and Singer, 2002]. This technique has been previously used for printed symbol recognition recognition with successful results [Malon et al., 2008].

Hidden Markov models

Hidden Markov models (HMMs) have been widely used for classification of online mathematical symbols [Garain and Chaudhuri, 2004; Hu and Zanibbi, 2011]. However their use in offline printed symbol recognition remained unexplored although HMMs have been successfully used for offline handwritten text recognition [Toselli et al., 2004a]. We applied the feature extraction process previously detailed for sequence classification with HMMs for this symbol recognition task. A brief description of HMMs is available in Section 3.3.4.

Chapter

4 Structural Analysis

A set of mathematical symbols has no meaning until relations are determined between them. The structure resulting from these relationships defines a mathematical expression. Automatic recognition of mathematical notation has two main problems: the recognition of the symbols, and the analysis of the structure of the expression. Although these problems can be considered independent, they are closely related. Within the formal framework defined in this thesis, we simultaneously optimized both problems by finding the most likely tree according to a structural model.

The structural analysis of mathematical expressions in this approach is driven by probabilistic grammars (2D-PCFG). However, the 2D-PCFG themselves are not enough to account for the structure of a given mathematical expression. Probability distributions that model the spatial relationships between symbols and subexpressions are also required.

In Section 4.1, we first introduce the content of this chapter regarding the structural analysis of mathematical expressions. The classification of spatial relationships between parts of an expression is described in Sections 4.2 and 4.3. Finally, we detail the estimation of the probabilistic grammar in Section 4.4.

Chapter Outline

4.1 Introduction

In the literature, there are several proposals for recognizing mathematical expressions. Many approaches have been reviewed in previous chapters according to the methodology proposed (see Section 2.1). In this thesis we defined an integrated approach based on parsing 2D-PCFG. The statistical framework stated the problem of recognizing math notation as finding the tree that maximizes the probability of Eq. (2.2.1). Also, we defined in Section 2.4 the structural probability of a given math expression as the parse tree obtained according to a 2D-PCFG.

The computation of the structural probability is defined by Eqs. (2.4.1) and (2.4.2). There are two main sources of information involved in the calculation of the probability of these equations. On the one hand, the spatial relationships probability between two regions: $p(r|BC)$. On the other hand, the productions of the 2D-PCFG whose probability has to be estimated: $p(a|A)$ and $p(BC|A)$. The definition of these distributions is provided below.

In order to provide examples and a more clear description, the definitions of this chapter are in terms of handwritten mathematics. However, as in other parts of this thesis given that we developed a general framework for mathematical expression recognition, the application to offline expressions is immediate. This is because features and classifiers are based on spatial information and pixels can be considered instead of points.

4.2 Spatial Relationships Classification

The computation of the structural probability of a math expression defined by Eq. (2.4.2) requires a spatial relationship model. This model has to provide the probability $p(r|BC)$ that two subproblems B and C are arranged according to spatial relationship r.

A common approach for obtaining a spatial relationship model is to define a set of geometric features to train a statistical classifier. Most proposals in the literature define geometric features based on the bounding boxes of the regions [Zanibbi et al., 2002; Awal et al., 2014]. The geometric features are usually modelled using Gaussian models [Awal et al., 2014], SVMs [Simistira et al., 2014] or fuzzy functions [Zhang et al., 2005], though some authors manually define specific functions [Zanibbi et al., 2002; Yamamoto et al., 2006; MacLean and Labahn, 2013]. All these approaches successfully deal with the classification of spatial relationships in math notation. Bounding boxes represent a good approximation for encoding regions and easily compute geometric

features, although in some cases other representations could be more useful for encoding the actual shapes of the symbols.

The spatial relations can be defined only between pairs of math symbols [Aly et al., 2009] or between symbols and subexpressions [Simistira et al., 2014]. Determining relationships only between symbols is less complex, but some structures could require special treatment because the relations may involve sets of symbols. For instance, before performing the structural analysis, Suzuki et al. [2003] decomposed each fraction into its numerator and denominator and extracted the expression in a root sign.

In this thesis, we cope with the classification of relationships between symbols and subexpressions. In order to recognize mathematical expressions, we consider five spatial relationships: *right* (BC), *below* ($\frac{B}{C}$), *subscript* (B_C), *superscript* (B^C) and *inside* ($\sqrt{C}$). These five relationships are enough because we process the expressions left-to-right and top-bottom.

In this section, we propose two sets of features for spatial relationship classification. Given two regions B and C, extracting a feature vector $\mathrm{h}(B, C)$ that represents their relationship we can train a classifier using labeled samples. Therefore, the probability of the spatial relationship model can be computed as the posterior probability provided by a statistical classifier (θ) for class r

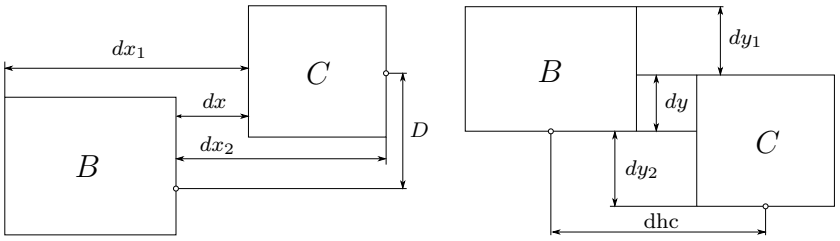
$$p(r \mid BC) = p_\theta(\, r \mid \mathrm{h}(B, C)\,)$$

Next we describe two sets of features for classifying spatial relationships based on two different representations. First, a feature set based on the bounding boxes of the regions. Second, a novel feature set based on the actual shape of the symbols. The evaluation of their performance on a public dataset is reported in Section 7.4.

4.2.1 Geometric Features: Bounding Boxes

Aly et al. [2009] presented two geometric features H and D for classifying spatial relationships between pairs of printed mathematical symbols. Based on their work, we extended the set of features with seven additional values. We aimed at improving classification results by providing more information given that handwritten expressions present more variability than printed expressions, and also because we considered relations between symbols and subexpressions. As a result, we defined nine geometric features for two given regions B and C based on their bounding boxes. This way, we compute the feature vector $\mathrm{h}(B, C)$ that represents their relationship and can be used for classification. The features are defined in Fig. 4.1, where H is the height of region

C, feature D is the difference between the vertical centroids, and 'dhc' is the difference between the horizontal centers. The features are normalized by a factor F such that we tested: the height of B, the combined height of regions B and C, or the distance between the centroids of regions B and C.



$$\mathrm{h}(B, C) = [H, D, \mathrm{dhc}, dx, dx_1, dx_2, dy, dy_1, dy_2]$$

Figure 4.1: Geometric features for classifying the spatial relationship between regions B and C.

The most challenging classification is between classes *right*, *subscript* and *superscript* [Zanibbi and Blostein, 2012]. An important feature for distinguishing between these three relationships is the difference between vertical centroids (D). Some symbols have ascenders, descenders or certain shapes such that the vertical centroid is not the best placement for the symbol center.

With a view to improving the placement of vertical centroids, we divided symbols into four typographic categories: ascendant (e.g. d or λ), descendant (p, μ), normal ($x, +$) and middle ($7, \Pi$). For *normal* symbols the centroid is set to the vertical centroid. For *ascendant* symbols the centroid is shifted downward to $(centroid + bottom)/2$. Likewise, for *descendant* symbols the centroid is shifted upward to $(centroid + top)/2$. Finally, for *middle* symbols the vertical centroid is defined as $(top + bottom)/2$. Fig. 4.2 shows an example of the different centers computed for a math symbol according to its type.

Once the vertical centers were calculated for every symbol, this information was hierarchically inherited as follows. The combination of two regions B and C resulting in a new region A had to follow some rules in order to preserve good center values. A decision was made depending on the spatial relation between them and the rule of the grammar, giving rise to different cases. In general, for the *subscript* relation (B_C) and the *superscript* relation (B^C) the center of their combined region A is the center of region B (Fig. 4.3a), for the *inside* relation ($\sqrt{C}$) the resulting center is the center of C (Fig. 4.3b), and for

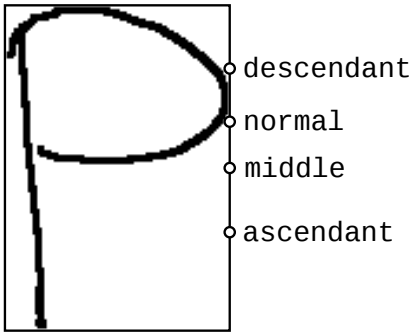


Figure 4.2: Example of vertical center computation for a mathematical symbol depending on its typographic category.

the *right* relation ($B\,C$) the final center is computed as the midpoint between their vertical centers (Fig. 4.3c). Finally, the *below* relation ($\frac{B}{C}$) depended on the rules of the grammar. For example, for parsing a fraction the following two rules could be used:

$$\text{Fraction} \xrightarrow{below} \text{Expression} \quad \text{OverExp}$$
$$\text{OverExp} \xrightarrow{below} \text{HorzLine} \quad \text{Expression}$$

where in the second rule the vertical center of 'OverExp' would be the center of 'HorzLine' and in the first rule the vertical center of 'Fraction' would be the center of 'OverExp'. As a result, the center of the region containing the fraction is the center its fraction line.

4.2.2 Shape Features: Polar Histograms

Many shape descriptors have been defined for image retrieval and object recognition in images [Yang et al., 2008]. Lately, shape-based features have been used in mathematical expression recognition to detect layout classes for symbols [Ouyang and Zanibbi, 2009], symbol retrieval [Marinai et al., 2011], symbol segmentation [Hu and Zanibbi, 2013], symbol recognition [Julca-Aguilar et al., 2014] and expression matching [Hirata and Honda, 2011]. But the application of shape-based features to spatial relation classification for math expressions remained unexplored.

In this section, we propose a novel set of features that is not based on bounding boxes but rather on the actual shape of the symbols. We define a shape-based feature set that is similar to shape contexts [Belongie et al., 2002;

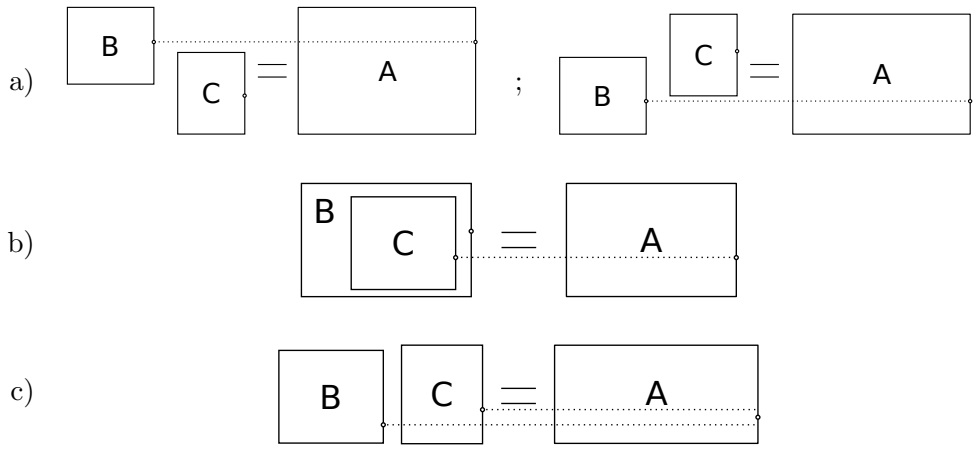


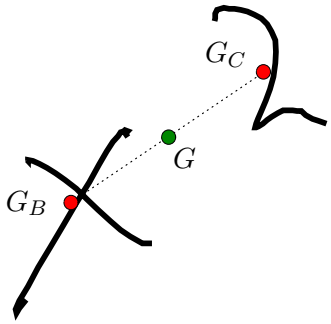
Figure 4.3: Vertical center computation of the combination of two regions according to the spatial relation among them: a) subscript and superscript; b) inside; c) right.

Yang et al., 2008]. We modify the polar shape matrix [Goshtasby, 1985], which provides a powerful descriptor that is invariant under translation, rotation and scaling. However, we wish to apply this descriptor to determine the relationship between two elements whose their relative position is important. As a result, we do not want rotation invariance.

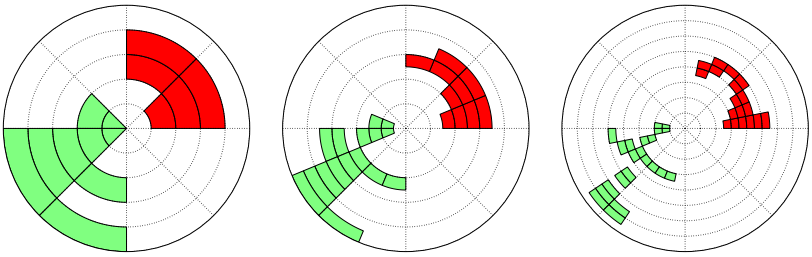
Given two sets of strokes B and C, let G_B and G_C be the center of mass of their corresponding shapes. Using $G = (G_B + G_C)/2$ as a center, we draw n circles with radii equally spaced up to the maximum radius containing B and C. Moving counterclockwise, draw radii dividing each circle into m equal arcs. This descriptor is encoded as a matrix $\mathcal{M}$ such that each row represents a circle and each column represent the angle starting from zero degrees. Each cell $\mathcal{M}(i, j)$ has one of three values obtained by majority vote of the points located in each bin:

$$\mathcal{M}(i,j) = \begin{cases} -1 & \text{more points from set } B \text{ than } C \\ 0 & \text{empty bin} \\ +1 & \textbf{tie}, \text{ or more points from set } C \text{ than } B \end{cases}$$

Figure 4.4 illustrates the effect of grid resolution on the polar histogram features. We see that as the grid size is increased, the representation is more detailed, producing a warped image of the strokes. The corresponding polar



Symbol pair (centers for x, 2 and midpoint shown).



$5 \times 8 = 40$ bins $\quad 10 \times 16 = 160$ bins $\quad 15 \times 32 = 480$ bins

Figure 4.4: Varying distance (n) $\times$ angle (m) resolution in a polar histogram layout descriptor. Values shown using green (-1), red (+1), and white (0).

shape matrix according to our features for $n = 5$ and $m = 8$ is

$$\begin{bmatrix} 0 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ +1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ +1 & +1 & 0 & 0 & -1 & -1 & 0 & 0 \\ +1 & +1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 0 \end{bmatrix}$$

Finally, the feature vector $h(B, C)$ describing the spatial relationship between the two regions is obtained by reducing the $n \times m$ features using Principal Component Analysis (PCA). Figure 4.5 illustrates the proposed descriptor for the five spatial relations considered.

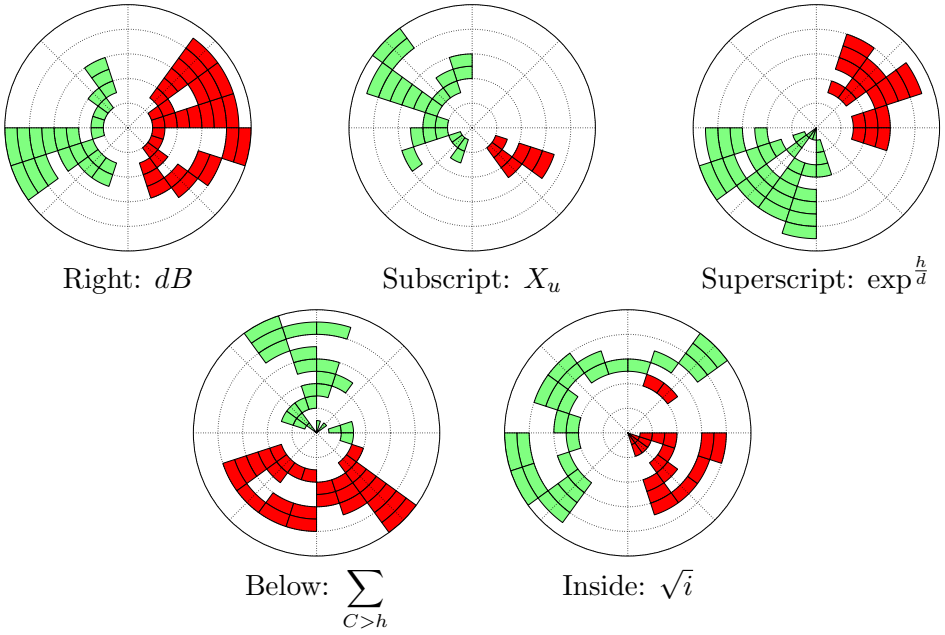


Figure 4.5: Polar histogram layout descriptors.

4.3 Clustering-based Penalty

The spatial relationship classifier is able to provide a probability for every relation r between any two given regions. However, there are several situations where we would not want the statistical model to assign the same probability as in other cases. Considering the expression in Fig. 4.6, the classifier might yield a high probability for *superscript* relationship '3^x', for the *below* relationship '$\frac{\pi}{2}$', and for the *right* relationship '2 3'; though we might expect a lower probability, since they are not the true relationships in the correct math expression.

Intuitively, those symbols or subexpressions that are closer together should be combined first. Furthermore, two symbols or subexpressions that are not *visible* from each other (see *visibility* definition in Section 3.2.2) should not be combined. These ideas are introduced into the spatial relationship model as a penalty based on the distance between strokes.

Specifically, given the combination of two hypotheses B and C, we computed a penalty function based on the minimum distance between the primi-

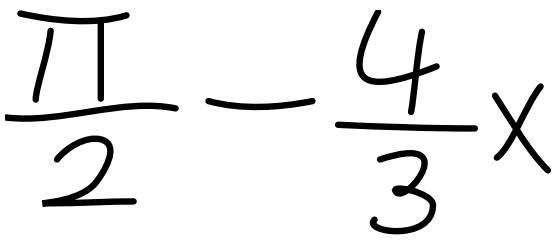


Figure 4.6: Example for hierarchical clustering penalty.

tives of B and C

$$\text{penalty}(B,C) = 1/(\ 1 + \min_{o_i \in B,\, o_j \in C} \text{d}(o_i, o_j)\ )$$

such that it is in the range $[0, 1]$. It should be noted that, although it is a penalty function, since it multiplies the probability of a hypothesis, the lower is the penalty value the more the probability is penalized.

This function is based on the single-linkage hierarchical clustering algorithm [Sibson, 1973] where, at each step, the two clusters separated by the shortest distance are combined. We defined a penalty function in order to avoid making hard decisions, because it is not always the case that the two closest strokes must be combined first.

The final statistical spatial relationship probability is computed as the product of the probability provided by the statistical classifier and the penalty function based on hierarchical clustering

$$p(r \mid BC) = p(r \mid \text{h}(B,C)) \cdot \text{penalty}(B,C) \tag{4.3.1}$$

An interesting property of the application of the penalty function is that, given that the distance between non-visible strokes is considered infinite, this function prunes many hypotheses. Furthermore, it favors the combination of closer strokes over strokes that are further apart. For example, the superscript relationship between symbols 3 and x in Fig. 4.6, although it could be likely, the penalty will favor that the 3 is first combined with the fraction bar, and later the fraction bar (and the entire fraction) with the x.

4.4 2D-PCFG Estimation

In the approach presented in this thesis, the computation of the structural probability of a mathematical expression required two statistical sources of in-

formation. The spatial relationships probability has been defined by Eq. (4.3.1). The other statistical source comes from the probabilistic grammar.

Since the rules of math notation are well-known, starting from the CFG provided by the organizers of the CROHME competition, we manually modified it to improve the modeling of some structures. We also added productions that increase ambiguity in order to model certain structures like the relations between categories of symbols (uppercase/lowercase letters, numbers, etc.). However, the probabilities of the productions of the 2D-PCFG have to be estimated.

The process for estimating the 2D-SCFG is detailed below, and the effect in performance of the estimation of the grammar is reported in the experimentation of Section 7.5 (see Tables 7.11 and 7.12).

4.4.1 Viterbi Estimation

An usual approach to estimating probabilistic grammars is the Viterbi score [Ney, 1992]. We recognized the expressions of the training set in order to obtain the most likely derivation trees according to the grammar. As recognizing the training set will introduce errors into the computed trees, we used constrained parsing to obtain the parse tree that best represents each training sample. Then, the probability of a production $A \to \alpha$ was calculated as

$$p(A \to \alpha) = \frac{c(A \to \alpha)}{c(A)}$$

such that $c(A \to \alpha)$ is the number of times that the rule $A \to \alpha$ was used when recognizing the training set, and $c(A)$ is the total number of productions used that have A as left-hand operator. In order to account for unseen events, we smoothed the probabilities using add-one smoothing [Jurafsky and Martin, 2008]. This smoothing method is naive and more advanced methodologies should lead to better results. However, we chose this method because it is simple and in the experimentation for validating the model, the domain of mathematical notation is known and the grammars only have a few hundreds of rules.

4.4.2 Constrained Parsing

Given a mathematical expression and its ground-truth information, we wanted to constrain the parsing process to recognize it perfectly with our model. There were two parts of the recognition process to take into account: symbol recognition and structural analysis.

First, symbol segmentation and classification is completely determined by the ground-truth, because the annotated information identifies each symbol (s_i) and the strokes that compose them (b_i). Thus, in the definition of the symbol likelihood of Eq. (2.3.1), we can set the true number of symbols K and the correct segmentations $b \in \mathcal{B}_K$. Also, for each symbol we consider that the probability resulting from the product of the three models (segmentation, classification and duration) is equal to one.

Regarding the structural analysis, it is not straightforward because several trees can represent the same mathematical expression. The ground-truth information of a mathematical expression commonly represents the structure as a tree [MacLean et al., 2011; Mouchère et al., 2013]. However, the most likely tree according to the 2D-PCFG of our parser can split the expression differently, producing another (equivalent) tree structure. For this reason, in order to guide the parser to recognize a given expression, we fed it with the list of spatial relationships provided in the ground-truth. Thus, the probability of Eq. (4.3.1) that two hypotheses B and C are arranged according to spatial relationship r, is set to one if that relationship is on the ground-truth (or can be inferred using transitive properties of relations). In this way, we favor the relationships extracted from the ground-truth while the parsing process is free to recognize other structure if the 2D-PCFG has a different construction.

Chapter

5 Parsing Mathematical Expressions

The fully integrated approach developed in this thesis for recognizing mathematical expressions is based on parsing 2D-PCFG. Throughout the previous chapters we have described in detail the statistical framework and the calculation of the different probabilistic sources required for computing the most likely expression. At this point, given an input expression, we need to define the procedure for recognizing the most likely mathematical expression according to our formal statistical framework.

In this chapter we describe the algorithm that computes the mathematical expression that maximizes the target probability of Eq. (2.2.1) according to the proposed approach. We also provide further analysis of the properties of the algorithm and discuss practical issues. The algorithm is described in terms of primitives such that it is applicable to any type of mathematical expression.

The parsing algorithm is presented in Section 5.1. Then, we analyze the complexity of the algorithm and some properties of the search space in Section 5.2. In Section 5.3 we discuss how the proposed approach deals with the recognition of symbols made up of multiple primitives. Finally, the process for training the system for math expression recognition is detailed in Section 5.4.

Chapter Outline

5.1 2D-PCFG Parsing Algorithm

In this section we present the algorithm for mathematical expression recognition that maximizes Eq. (2.2.1). We define a CYK-based algorithm for parsing 2D-PCFGs in the statistical framework described in this thesis. Using this algorithm, we compute the most likely parse tree according to the proposed model for a given input $\boldsymbol{o}$ containing N primitives.

The parsing algorithm is essentially a dynamic programming method. First, the initialization step computes the probability of several mathematical symbols for each possible segmentation hypothesis. Second, the general case computes the probability of combining different hypotheses such that it builds the structure of the mathematical expression.

The dynamic programming algorithm computes a probabilistic parse table γ. Following a notation similar to [Goodman, 1999], each element of γ is a probabilistic nonterminal vector, where their components are defined as:

$$\gamma(A, b, l) \;=\; \hat{p}(A \overset{*}{\Rightarrow} b); \qquad l \;=\; |\, b \,|$$

such that $\gamma(A, b, l)$ denotes the probability of the best derivation where nonterminal A generates a set of primitives b of size l.

Initialization. In this step the parsing algorithm computes the probability of every admissible segmentation $b \in \mathcal{B}$ as described in Section 3.2.2. The probability of each segmentation hypothesis is computed according to Eqs. (2.2.1) and (2.3.1) as

$$\gamma(A, b_i, l) = \max_{s} \{\, p(s \mid A)\; p(b_i)\; p(s \mid b_i)\; p(l \mid s) \,\} \tag{5.1.1}$$

$$\forall A, \forall K,\ \forall b \in \mathcal{B}_K,\ 1 \leq i \leq |b|,\ 1 \leq l \leq \min(N, L_{\max})$$

where $L_{\max}$ is a parameter that constrains the maximum number of primitives that a symbol can have.

This probability is the product of a range of factors such that it is maximized for every mathematical symbol class s: probability of terminal rule, $p(s|A)$ (Eq. (2.4.1)), probability of segmentation model, $p(b_i)$ (Eq. (3.2.1)), probability of mathematical symbol classifier, $p(s|b_i)$ (Eq. (3.3.1)), and probability of duration model, $p(l_i|s)$ (Eq. (3.1.1)).

General case. In this step the parsing algorithm computes a new hypothesis $\gamma(A, b, l)$ by merging previously computed hypotheses from the parsing table

until all N primitives are parsed. The probability of each new hypothesis is calculated according to Eqs. (2.2.1) and (2.4.2) as:

$$\begin{aligned}\gamma(A,b,l) = \max\{\ \gamma(A,b,l),\ \max_{B,C}\ \max_{r}\ \max_{b_B,b_C}\ \{ \\ p(BC \mid A)\gamma(B,b_B,l_B)\,\gamma(C,b_C,l_C)\,p(r \mid BC)\ \}\} \\ \forall A,\ 2 \le l \le N\end{aligned} \tag{5.1.2}$$

such that $b = b_B \cup b_C$; $b_B \cap b_C = \emptyset$ and $l = l_B + l_C$.

This expression shows how a new hypothesis $\gamma(A,b,l)$ is built by combining two subproblems $\gamma(B,b_B,l_B)$ and $\gamma(C,b_C,l_C)$, considering both syntactic and spatial information: probability of binary grammar rule $p(BC|A)$ (Eq. (2.4.2)) and probability of spatial relationship classifier $p(r|BC)$ (Eq. (4.3.1)). It should be noted that both distributions significantly reduce the number of hypotheses that are merged. Also, the probability is maximized taking into account that a probability might already have been set by the Eq. (5.1.1) during the initialization step.

Finally, the most likely hypothesis and its associated derivation tree $\hat{t}$ that accounts for the input expression can be retrieved in $\gamma(S,\boldsymbol{o},N)$ (where S is the start symbol of the grammar). The pseudocode of the algorithm for parsing mathematical expressions is provided in Algorithm 5.1.

5.2 Complexity and Search Space

We have defined an integrated approach for mathematical expression recognition based on parsing 2D-PCFG. The dynamic programming algorithm is defined by the corresponding recursive equations. The initialization step is performed by Eq. (5.1.1), while the general case is computed according to Eq. (5.1.2). We have also presented the pseudocode of the parsing algorithm in Algorithm 5.1. In addition to the formal definition, there are some details of the parsing algorithm regarding the search space that need further explanation.

The initialization step computes the probability for every admissible segmentation hypothesis ($\mathcal{B}$) limited by the parameter $L_{\max}$. After the initialization step, the general case is the core of the algorithm. The first loop determines the size l_A of the hypotheses to be built. Then, the second loop generates all the sizes to split l_A primitives into $l_B + l_C = l_A$. Once the sizes are set, for every set of primitives b_B we try to create a new hypothesis for every possible combination with another set b_C using the binary rules of the grammar. According to this algorithm we can see that the time complexity for

Algorithm 5.1: CYK for parsing math expressions

input : $\mathcal{G}$ (2D-PCFG grammar with start symbol S)
$\boldsymbol{o}$ (sequence of N primitives)
$L_{\max}$ (max number of primitives per symbol)
output: most likely parse tree $\hat{t}$ that maximizes Eq. (2.2.1)
begin

// Initialization step
for $l = 1$ *to* $\min(L_{\max}, N)$ **do**
  for $b_i \in \mathcal{B} : |b_i| = l$ **do**
    for $A \to s \in \mathcal{G}$ **do**
      $\text{pr} = p(s \mid A)\; p(b_i)\; p(s \mid b_i)\; p(l \mid s)$
      if $\text{pr} > 0.0$ *and* $\text{pr} > \gamma(A, b_i, l).\text{p}$ **then**
        $\gamma(A, b_i, l).\text{p} \;\; = \text{pr}$
        $\gamma(A, b_i, l).\text{rule} = A \to s$

// General case
for $l_A = 2$ *to* N **do**
  for $l_B = 1$ *to* $l_A - 1$ **do**
    $l_C = l_A - l_B$
    for $b_B : \; \exists\gamma(B, b_B, l_B)$ **do**
      for $b_C : \; \exists\gamma(C, b_C, l_C)$ **do**
        for $A \xrightarrow{r} BC \in \mathcal{G}$ **do**
          $\text{pr} = p(BC|A)\, p(r|BC)\gamma(B, b_B, l_B).p\; \gamma(C, b_C, l_C).p$
          if $\text{pr} > 0.0$ *and* $\text{pr} > \gamma(A, b_A, l_A).\text{p}$ **then**
            $\gamma(A, b_A, l_A).\text{p} \;\; = \text{pr}$
            $\gamma(A, b_A, l_A).\text{rule} \; = A \xrightarrow{r} BC$
            $\gamma(A, b_A, l_A).\text{left} \; = \gamma(B, b_B, l_B)$
            $\gamma(A, b_A, l_A).\text{right} = \gamma(C, b_C, l_C)$

// The most likely hypothesis that accounts for the input $\boldsymbol{o}$
return $\gamma(S, \boldsymbol{o}, N)$

parsing an input expression of N primitives is $O(N^4|P|)$ where $|P|$ is the number of productions of the grammar. However, this complexity can be reduced by constraining the search space.

The intuitive idea is that, given a set of primitives b_B, we do not need to try to combine it with every other set b_C. A set of primitives b_B defines a region in space, allowing us to limit the set of hypothesis b_C to those that fall within a region of interest. For example, given symbol 4 in Fig. 4.6, we only have to check for combinations with the fraction bar and symbol 3 (below relationship) and the symbol x (right or sub/superscript relationships).

We applied this idea using the region in the two-dimensional space associated with a set of primitives.

Definition 5.2.1. Given a primitive o_i, its associated region $r(o_i) = (x, y, s, t)$ in the 2D space is the minimum bounding box that contains that primitive, where (x, y) is the top-left coordinate and (s, t) the bottom-right coordinate of the region.

Definition 5.2.2. Likewise, given a set of primitives $b = \{o_j \mid 1 \leq j \leq |b|\}$, its associated region $r(b) = (x_b, y_b, s_b, t_b)$ is the minimum rectangle that contains all primitives $o_j \in b$.

Therefore, given a spatial region $r(b_B)$ we retrieve only the hypotheses b_C whose region $r(b_C)$ falls in a given area $\mathcal{R}$ relative to $r(b_B)$. Fig. 5.1 shows the definition of the regions in the space in order to retrieve relevant hypotheses to combine with b_B depending on the spatial relation.

The dimensions of the *normalized symbol size* (R_w, R_h) are computed as the maximum between the average and median width of the input primitives (R_w); and the maximum between the average and median height of the input primitives (R_h). These calculations are independent of the input resolution. The normalized symbol size is also used to normalize other distance-related metrics in the model, like determining what primitives are close together in the multi-primitives symbol recognition or the normalization factor of features in the segmentation model.

In order to efficiently retrieve the hypotheses falling in a given region $\mathcal{R}$, every time a set of hypotheses of size l_A is computed, we sort this set according to the x coordinate of every region $r(b_A)$ associated with $\gamma(A, b_A, l_A)$. This sorting operation has cost $O(N \log N)$. Afterwards, given a rectangle $r(b_B)$ in the search space and a given size l_C, we can retrieve the hypotheses $\gamma(C, b_C, l_C)$ falling within that area by performing a binary search over that set in $O(\log N)$. Although the regions are arranged in two-dimensions and

they are sorted only in one dimension, this approach is reasonable since math expressions grow mainly from left to right.

Assuming that this binary search will retrieve a small constant number of hypothesis, the final complexity achieved is $O(N^3 \log N|P|)$. Furthermore, many unlikely hypotheses are pruned during the parsing process.

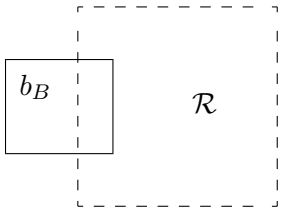
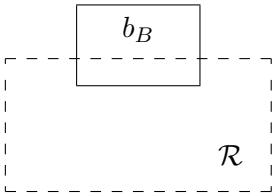
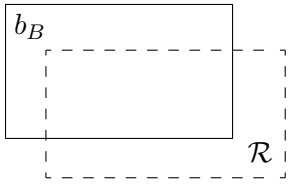
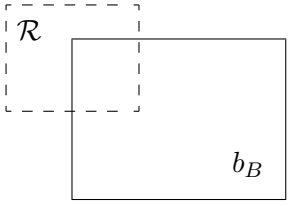
Right, Subscript, Superscript $x = \max(r(b_B).x + 1, r(b_B).s - R_w)$ $y = r(b_B).y - R_h$ $s = r(b_B).s + 8R_w$ $t = r(b_B).t + R_h$	
Below $x = r(b_B).x - 2R_w$ $y = \max(r(b_B).y + 1, r(b_B).t - R_h)$ $s = r(b_B).s + 2R_w$ $t = r(b_B).t + 3R_h$	
Inside $x = r(b_B).x + 1$ $y = r(b_B).y + 1$ $s = r(b_B).s + R_w$ $t = r(b_B).t + R_h$	
Mroot $x = r(b_B).x - 2R_w$ $y = r(b_B).y - R_h$ $s = \min(r(b_B).s, r(b_B).x + 2R_w)$ $t = \min(r(b_B).t, r(b_B).y + 2R_h)$	

Figure 5.1: Spatial regions defined to retrieve hypotheses relative to hypothesis b_B according to different relations.

5.3 Multi-primitive Symbol Recognition

Detecting symbols composed of multiple primitives is a challenging task in mathematical expression recognition. The parsing algorithm developed in this thesis for recognizing mathematical expressions has two steps. First, when the parsing table is initialized, several symbol hypotheses are introduced as subproblems of $1 \leq n \leq L_{\max}$ primitives following Eq. (5.1.1). Then, in the general case, the parsing table is filled with hypotheses of $n \geq 2$ primitives by combining smaller subproblems using Eq. (5.1.2). This produces a scaling problem of the probabilities.

The probability of a multi-primitive symbol hypothesis of size $n \geq 2$ created in the initialization step is always the product of four probabilities (Eq. (5.1.1)), while the hypothesis resulting from the combination of subproblems will involve $6n - 2$ terms in the calculation. For instance, when two hypotheses are combined using the Equation (5.1.2), the resulting probability of a hypothesis of size $l_A = 2$ is

$$\gamma(A, b_A, 2) = p(BC \mid A)\, \gamma(B, b_B, 1)\, \gamma(C, b_C, 1)\, p(r \mid BC)$$

such that is the result of multiplying ten probabilities: $p(BC|A)$, $p(r|BC)$ and twice the four probabilistic models of each hypothesis of size one. Thus, the model would favor the multi-primitive hypotheses at the sacrifice of combining smaller symbols. In general, parsing PCFG is biased to consider smaller trees more probable [Manning and Schütze, 1999].

In addition to this scaling problem, the different probabilistic distributions have been estimated separately, which leads to values in different scales. For these reasons, following [Luo et al., 2008], we assigned different exponential weights (w_i) to each model probability, and we also added an insertion penalty (I) in the initialization step (Eq. (5.1.1)). Taking into account these decisions, the equation of the parsing algorithm for the initialization step becomes

$$\gamma(A, b_i, l) = \max_{s} \{\, I\; p(s \mid A)^{w_1}\; p(b_i)^{w_2}\; p(s \mid b_i)^{w_3}\; p(l \mid s)^{w_4} \,\}$$

$$\forall A, \forall K,\ \forall b \in \mathcal{B}_K,\ 1 \leq i \leq |b|,\ 1 \leq l \leq \min(N, L_{\max})$$

and the equation of the general case becomes

$$\begin{aligned}\gamma(A, b, l) = \max\{\, \gamma(A, b, l),\ \max_{B,C}\ \max_{r}\ \max_{b_B, b_C} \{ \\ p(BC \mid A)^{w_5} \gamma(B, b_B, l_B)\, \gamma(C, b_C, l_C)\, p(r \mid BC)^{w_6} \,\}\} \\ \forall A,\ 2 \leq l \leq N\end{aligned}$$

These parameters alleviate the scaling differences of the probabilities. Furthermore, the weights help to adjust the contribution of each model to the final probability, since some sources of information are more relevant than the others. The parameters can be tuned, for instance, minimizing an error metric using a development dataset.

Additionally, there are some multi-primitive mathematical symbols that can be naturally recognized by adding specific rules to the grammar. For example, the following rule

$$\text{Equals}(=) \xrightarrow{\text{below}} \text{Hline}(-) \quad \text{Hline}(-)$$

models an equals sign as the vertical concatenation of two horizontal lines. Other examples of symbols that can be detected using grammar rules are: $\pm$, $\leq$ or cos. Therefore, the hypotheses generated at the initialization step of the parsing process along with these specific rules provide a powerful method for recognizing multi-primitive mathematical symbols, where symbol segmentation becomes a hidden variable.

5.4 Training Process

A mathematical expression recognition system that implements the parsing algorithm described in this chapter requires some steps to be ready for recognition. As discussed in this thesis, there are problems at different levels. Given an input sample, we developed an integrated approach that globally computes the most likely expression, but several probabilistic sources and parameters are used in the calculation. In this section we explain the process for training the math expression recognition system using a set of annotated math expression samples for training, and another set for validation. The training process is divided in two stages.

First, we need to estimate the distributions required for computing the probabilities of the hypotheses generated during the parsing process: the symbol likelihood and the structural probability. Fig. 5.2 shows a diagram of the steps required to train an initial system for recognizing mathematical expressions, which involves the following tasks:

- For each annotated math expression in the train set, we extract samples of: mathematical symbols, wrong symbol segmentations and spatial relationships. It is important to extract hypotheses taking into account the constraints used during the recognition algorithm: the admissible segmentation hypotheses (Section 3.2) and the considered spatial relationships (Section 4.2).

- The calculation of the **symbol likelihood** requires the following estimations:
  - Counting the number of primitives for each symbol sample in the train set, we estimate the **symbol duration model** (Section 3.1).
  - Using the mathematical symbol samples and the wrong segmentation samples, we extract the geometric features for segmentation described in Section 3.2.2 and we train a statistical classifier. This classifier provides the probability of the **symbol segmentation model**.
  - As described in Chapter 3, the **symbol classification model** is obtained as the posterior probability provided by a statistical classifier using Eq. (3.3.1). Using the symbol samples from the train set, we extract the specific features and train the corresponding classifiers depending on the type of expression: handwritten expressions (Section 3.3) or printed expressions (Section 3.4).
- The **structural probability** requires the following steps:
  - The samples of spatial relationships extracted from the train set of mathematical expressions are used to compute several feature vectors of different relationships (Section 4.2). The probability of the **spatial relationships model** is obtained as the posterior probability of a statistical classifier trained with the relationships encoded as feature vectors.
  - A **2D-PCFG** is required for computing the structural probability. Since the rules of math notation are well-known, a grammar can be manually constructed considering all productions equiprobable. Also, the organizers of the CROHME competition provided a CFG that can be easily converted into a 2D-PCFG.
- Finally, all the estimated models along with a 2D-PCFG, are the resources necessary to compute the required probabilities and build our initial system for mathematical expression recognition.

The initial system for mathematical expression recognition is able to parse input expressions, but the parameters of the system (segmentation distance threshold, insertion penalty and exponential weights) have to be tuned in order to obtain the best performance of the proposed approach. Also, the probabilities of the productions of the 2D-PCFG have to be estimated.

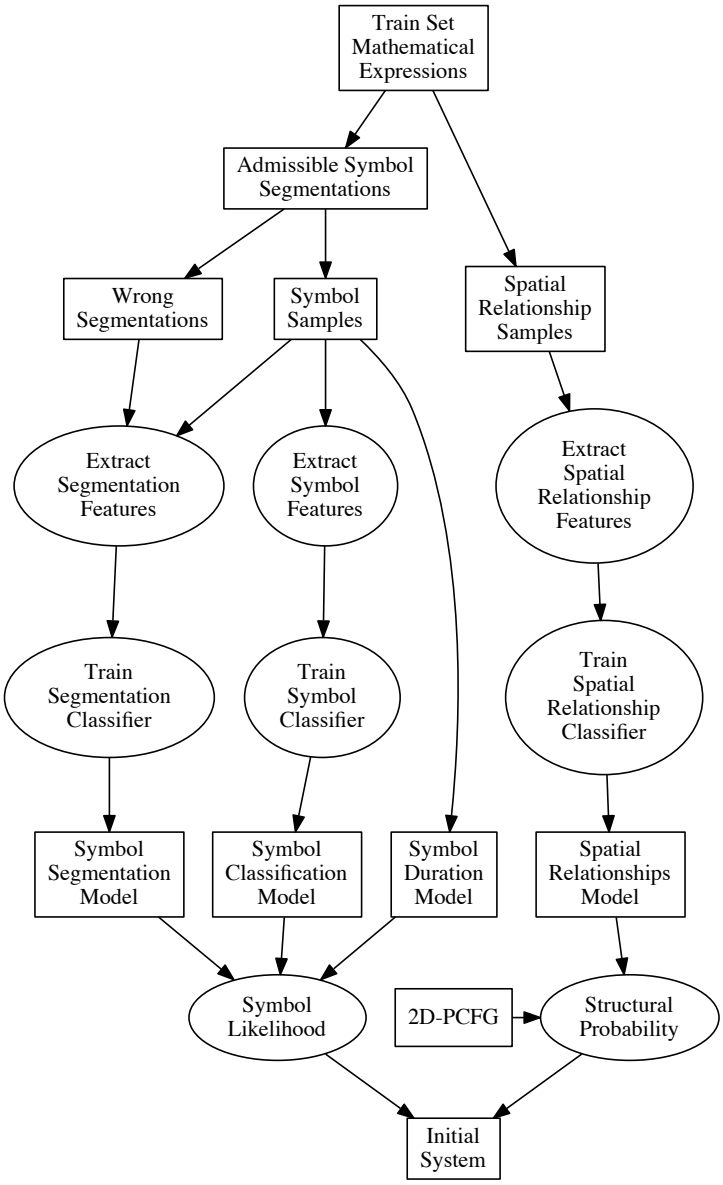


Figure 5.2: Diagram of the process for training the initial mathematical expression recognition system.

The second stage for training the final recognition system is described in Fig. 5.3, requiring the following tasks:

- The initial system is built with the probability distributions previously estimated and all parameters set initially to one ($I = b_{\text{th}} = w_i = 1.0$). The maximum number of primitives that a symbol can have ($L_{\max}$) can be set manually in order to accounts for most of the symbols. The initial system with this configuration is used to recognize the validation set. Then, we tune the parameters using the Downhill Simplex algorithm [Nelder and Mead, 1965] such that a performance error metric is minimized. As a result, we obtain the initial system with the parameters tuned.

- The next step is estimating the probabilities of the productions of the grammar (see Section 4.4). To this end, we recognize the expressions in the train set using the tuned system and constrained parsing. Then, we estimate the probabilities of the 2D-PCFG using the Viterbi score with the set of parse trees from the training samples.

- Finally, we repeat the parameter tuning procedure minimizing a performance error metric when recognizing the validation set, but this time using the estimated 2D-PCFG. The estimated grammar and probabilistic models, along with the parameters tuned in this step, are the final configuration for the mathematical expression recognition system.

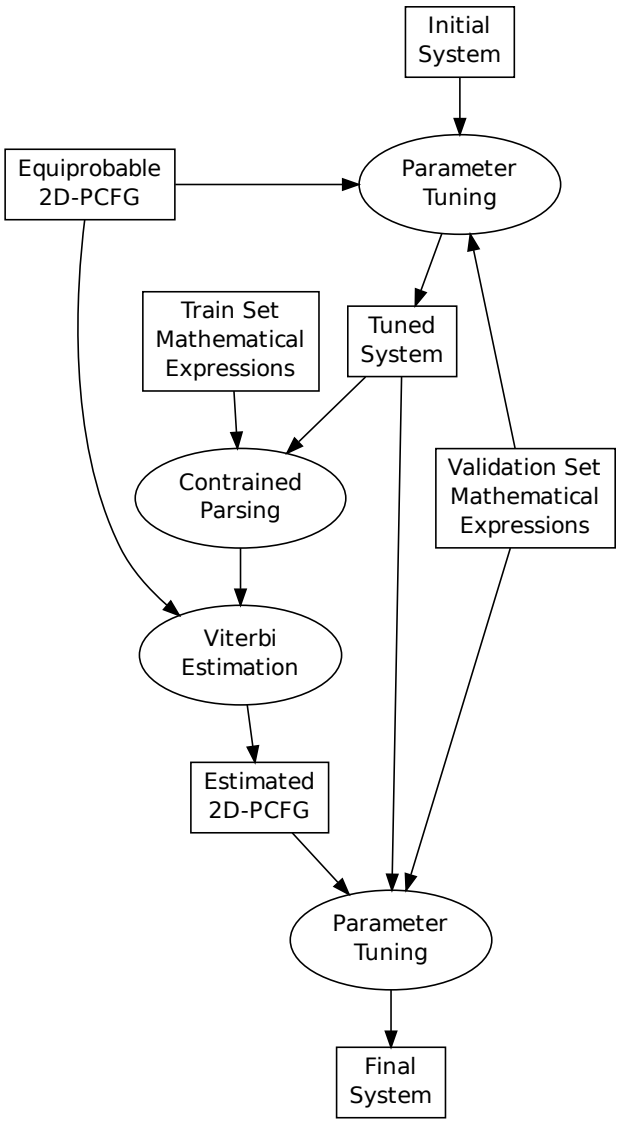


Figure 5.3: Diagram of the process for training the final mathematical expression recognition system.

Chapter

6 The Problem of Performance Evaluation

Assessing the performance of different solutions to a problem is crucial in order to evaluate the advancements in a specific field such that research can move towards the best approach to deal with it. A good set of performance metrics along with large public datasets is the desired scenario for comparing different approaches and help the research community. Unbiased metrics that can be computed automatically are very important for objective evaluation. Furthermore, in many pattern recognition problems is common to estimate the parameters of a model by minimizing an error function based on a certain metric.

Automatic performance evaluation in mathematical expression recognition is not straightforward. There are several issues that have made comparison in this field difficult. In this chapter we discuss the problems related with the assessment of mathematical expression recognition experiments and the solutions proposed so far.

The chapter is organized as follows. In Section 6.1 we describe and discuss the main issues related to automatic evaluation of mathematical expression recognition. Then, in Section 6.2 we present and analyze different metrics used in the literature for performance evaluation, where we developed and proposed a new metric. Finally, in Section 6.3 we discuss the problems and solutions presented in this chapter.

Chapter Outline

6.1 Introduction

Diverse issues arise in performance evaluation of mathematical expression recognition systems. A deep discussion about this problem was provided by Lapointe and Blostein [2009] and Awal et al. [2010]. Next, we review the main problems that make automatic performance evaluation difficult in this field.

One of the main issues in performance evaluation of math notation is that there are many ambiguities at different levels. First, there are ambiguities inherent to the expressions that accept different interpretations. Awal et al. [2010] show some examples like the expression $f(y+1)$ that can be considered as the variable f multiplying the term $(y+1)$, or the function f applied to the value $y+1$; or the expression $a/2b$ that can be interpreted as a fraction with denominator $2b$ or the product between the fraction $a/2$ and the variable b. Other ambiguities are due to handwriting production. In Section 1.2 there are several examples of ambiguities at different levels, like Fig. 1.4 shows ambiguous symbol segmentations and Fig. 1.6 presents different interpretations of the same shapes.

These previous sources of ambiguity produce that more than one ground-truth could be valid for a given expression. Nevertheless, even if a math expression is not ambiguous, the representation formats do not enforce uniqueness [Lapointe and Blostein, 2009]. Math expressions are usually encoded in LaTeX or MathML, where the same expression can be annotated by several correct representations as shown in Fig. 6.1. All the described ambiguities can produce that a correct recognition result for a given expression does not match the ground-truth, therefore reporting undesired recognition errors. Consequently, metrics for automatic performance evaluation of math expression recognition should be based on formats that specify a unique encoding for a given math expression.

Commonly, mathematical expression recognition is divided in three different problems (see Section 1.2): segmentation, symbol recognition and structural analysis. Although several issues have been described, symbol segmentation and symbol recognition can be easily calculated. The only remaining ambiguity is the interpretation of an expression. Measuring errors in the structure of the expression is the most challenging task. Many authors report symbol segmentation rate, symbol recognition rate and the expression recognition rate. However, the expression recognition rate is hard to be automated due to the representation ambiguities, thus several results are computed manually [Awal et al., 2010]. Moreover, it is a very pessimistic metric such that a single error in an expression produces a wrong recognition result. Global

LaTeX	MathML	
x_a^2 + 1 x_a^{2} + 1 x_{a}^2 + 1 x_{a}^{2} + 1 x^2_a + 1 x^2_{a} + 1 x^{2}_a + 1 x^{2}_{a} + 1	<mathml> <mrow> <msubsup> <mi>x</mi> <mi>a</mi> <mn>2</mn> </msubsup> <mo>+</mo> <mn>1</mn> </mrow> </mathml>	<mathml> <mrow> <msubsup> <mi>x</mi> <mi>a</mi> <mn>2</mn> </msubsup> <mrow> <mo>+</mo> <mn>1</mn> </mrow> </mrow> </mathml>

Figure 6.1: Some examples of different valid representations for math expression $\boxed{x_a^2 + 1}$ in LaTeX and MathML format.

error values can be computed as an edit distance between strings or trees, but the encoding of the math expressions has to deal with the representation ambiguities. Furthermore, edit distances report a global error (frequently not normalized), such that the source of the error is unknown (segmentation, symbols, structure). Several proposals of metrics for math expression recognition evaluation are detailed in next Section 6.2.

Despite the problems previously described, there are other issues that make comparison of approaches difficult. For many years, in the literature several methods and techniques were proposed, but there was a shortage of large, representative, publicly available ground-truthed datasets [Lapointe and Blostein, 2009]. As a result, many approaches reported results on small private datasets collected by the authors, such that comparison is not feasible.

A good way to provide comparable results is being able to reproduce a reported experimentation. If the software used in a given approach is made available and the parameters and procedures are clearly described, further proposals could be compared. Unfortunately, the systems are not usually released, and often the experimentation is not detailed enough to allow performing the reported experiments using the same parameter settings.

Fortunately, mathematical expression recognition field has been tackling all these problems. During the last years, several public datasets have been

released (see Section 7.1), many different metrics have been proposed (see next Section 6.2) and even the CROHME international competition is held every year providing results of various approaches under the same conditions.

6.2 Performance Evaluation Metrics

In previous section we have introduced and discussed the main issues that make difficult the automatic performance evaluation of mathematical expression recognition. In this section we describe the metrics that have been proposed to this end, analyzing their strengths and weaknesses.

6.2.1 Early Global Metrics

Expression recognition rate is a metric for computing the overall performance of a math expression recognition system. It is commonly reported along with other metrics at symbol level [Okamoto et al., 2001; Zanibbi et al., 2002]. Recognition rate at expression level complements the symbol level evaluation, but it is a pessimistic metric because a single error causes the entire expression to be a wrong recognition result. Furthermore, its computation has to deal with representation ambiguities.

Later, other global metrics were proposed as a combination of recognition rates at different levels. Chan and Yeung [2001] proposed an *integrated performance measure* as the ratio of the number of correctly recognized symbols and operators (structure) to the total number of symbols and operators tested. Garain and Chaudhuri [2005a] defined a *global performance index* that combines the number of symbols recognized incorrectly and the number of symbols incorrectly arranged in the expression. They also penalized differently the structural errors depending on the *level* of the symbol, such that the dominant baseline of an expression is treated as level zero and the level number increases above and decreases below the baseline.

These first proposals for computing a global error integrate the errors at symbol level and at structure level. However, segmentation errors are not taken into account and would affect the computation of these metrics because not direct matching could be possible between expressions. Also, determining implicit operators in the *integrated performance measure* or the incorrect arrangements in levels of the *global performance index* is not straightforward, and the software for evaluation was not made available.

6.2.2 EMERS

A mathematical expression can be naturally represented as a tree (see Fig. 1.8). The tree representation, commonly in MathML format, contains simultaneously the symbols and the structure of a given mathematical expression. For this reason, computing an edit distance between trees is an appropriate method in order to compute the error between a recognized expression and its ground-truth tree.

Sain et al. [2010] proposed EMERS[a], a tree matching-based performance evaluation metric for mathematical expression recognition. Using the tree representation of two expressions in MathML (which can also be easily obtained from LaTeX) they defined a method for computing the edit distance between them. Since matching of trees is a hard problem, they proposed to match ordered trees represented by their corresponding Euler strings. Given two trees encoded by two Euler strings A and B, the overall complexity of the EMERS algorithm is $O(|A|^2|B|^2)$ or more generally $O(n^4)$.

EMERS computes the set of edit operations that transform the recognized tree into the ground-truth tree. Accordingly, EMERS is not a normalized metric but an edit distance, such that if both trees are identical EMERS is equal to zero. The edit distance between trees is a well-defined metric but the representation ambiguity of MathML can produce that correct recognition results are considered errors. We performed an experiment comparing the trees of a recognition experiment with the ground-truth trees, and also comparing the recognized trees with the equivalent trees the system would provide in an error-free experiment (using constrained parsing as described in Section 4.4.2). The expression recognition rate, computed as the percentage of expressions with EMERS equals to zero, differed by almost 8% depending on the ground-truth used [Álvaro et al., 2012]. A canonical form to represent math expressions in MathML should be required in order to avoid this problem. Sain et al. [2010] try to overcome this problem by converting the MathML to LaTeX and then converting the LaTeX back to MathML.

Furthermore, the edit operations can be for symbols or MathML tags, such that the distance between some symbols is different depending on its type (`<mi>`, `<mo>`, `<mn>`). For example, the distance between letter s and symbol x would be one (modify s to x), whereas the distance between letter s and number 5 would be two (modify `<mi>` to `<mn>` and modify s to 5). As other global metrics, the computed error value accounts for the entire expression but the source of the errors is not explicitly known. The set of edit operations

[a] Available at http://www.isical.ac.in/~utpal/resources.php

is provided and we could compute if they are related to symbols or tags, but, for instance, segmentation mistakes could not be detected and would become symbol and tags errors.

Finally, the authors propose two options for computing the error: every edit operation has the same cost, or it depends on the baseline (using the concept of *level* defined in previous section) in which the edit operators are done. The default EMERS value is computed using the weighted version, and this produces that the distance is not symmetric in some cases.

6.2.3 IMEGE

While researching on mathematical expression recognition, automatic performance evaluation metrics are very important in order to assess if new strategies are improving the recognition performance. Expression recognition rate was pessimistic and had the problem of representation ambiguity. For these reasons, we developed a novel performance evaluation metric trying to overcome these problems.

Given a recognition result of a certain expression and its ground-truth we wanted to evaluate the quality of this result. The image representation of a math expression can be generated from its string codification (e.g. LaTeX or MathML). Since there can be several string representation of the same expression but the image obtained should be unique, we proposed comparing images directly to compute an error metric. Next we explain how by using an image-matching model (IDM), we defined the evaluation algorithm (BIDM) that is used to finally compute the recognition error (IMEGE).

Image-matching model (IDM)

In order to obtain a matching between two images, the initial idea was to compute a two-dimensional warping between them. Keysers et al. [2007] presented several deformation models for image classification, and the Image Distortion Model (IDM) represented the best compromise between computational complexity and evaluation accuracy. For this reason, we chose the IDM to perform a two-dimensional matching between two images.

The IDM is a zero-order model of image variability [Keysers et al., 2007]. This model uses a mapping function with absolute constraints; hence, it is computationally much simpler than a two-dimensional warping. Its lack of constraints is compensated using a local gradient image context window. This model obtains a dissimilitude measure from one image to another such that if two images are identical, their distance is equal to zero.

The IDM has two parameters: warp range (w) and context window size (c). The algorithm requires each pixel in the test image to be mapped to a pixel within the reference image not more than w pixels from the place it would take in a linear matching. Over all these possible mappings, the best matching pixel is determined using the $c \times c$ local gradient context window by minimizing the difference with the test image pixel. Fig. 6.2 illustrates how the IDM works and the contribution of both parameters, where the warp range w constrains the set of possible mappings and the $c \times c$ context window computes the difference between the horizontal and vertical derivatives for each mapping. It should be noted that these parameters need to be tuned.

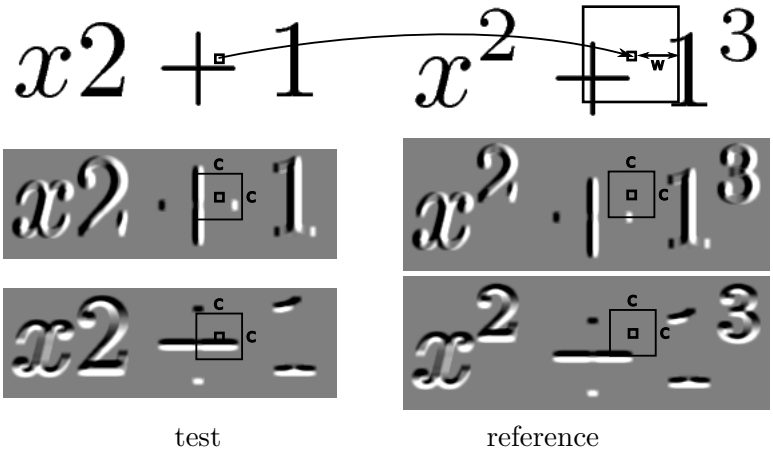


Figure 6.2: Image Distortion Model (IDM) visual representation.

The evaluation algorithm (BIDM)

Once we had a model that was able to detect similar regions of two images, we wanted to use this information to compute an error measure between them. Starting from the IDM-distance algorithm presented in [Keysers et al., 2007], we proposed the Binary IDM (BIDM) evaluation algorithm (defined in Algorithm. 6.1). First, instead of calculating the vertical and horizontal derivatives using Sobel filters, these derivatives are computed using the method described in [Toselli et al., 2004a]. Next, the double loop computes the IDM distance for each pixel, and these values are stored individually. After that, the difference between each pixel of the test image and the most similar pixel found in the reference image can be represented as a gray-scale image (Fig. 6.3c-1). At this

Algorithm 6.1: Binary IDM (BIDM) evaluation algorithm.

input : test image A $(I \times J)$
reference image B $(X \times Y)$
warp range w
context window size c
output: BIDM(w, c) from A to B
begin

$A^v = \text{vertical\_derivative}(A)$
$A^h = \text{horizontal\_derivative}(A)$
$B^v = \text{vertical\_derivative}(B)$
$B^h = \text{horizontal\_derivative}(B)$

for $i = 1$ *to* I **do**
  for $j = 1$ *to* J **do**
    $i' = \left\lfloor i\frac{X}{I} \right\rfloor$, $j' = \left\lfloor j\frac{Y}{J} \right\rfloor$, $z = \left\lfloor \frac{c}{2} \right\rfloor$;
    $S_1 = \{1, \ldots, X\} \cap \{i' - w, \ldots, i' + w\}$;
    $S_2 = \{1, \ldots, Y\} \cap \{j' - w, \ldots, j' + w\}$;

$$\text{map}(i,j) = \min_{\substack{x \in S_1 \\ y \in S_2}} \sum_{m=-z}^{z} \sum_{n=-z}^{z} (A^v_{i+n,j+m} - B^v_{x+n,y+m})^2 + (A^h_{i+n,j+m} - B^h_{x+n,y+m})^2$$

normalize_depth(map, 255)
binarize(map) *//Otsu's method*

fg $= \{(x, y) \mid A(x, y) < 255\}$ *//Foreground pixels*
cp $=$ fg $\cap \{(x, y) \mid map(x, y) = 0\}$ *//Correct pixels*

return $\frac{|cp|}{|fg|}$ *//Correct pixels ratio*

point, we have a dissimilitude value for each pixel of the test image. However, rather than knowing how different a pixels is, we want to know whether or not a pixel is correct. This is achieved by normalizing the distance values in the range $[0, 255]$ and then performing a binarization process using Otsu's method [Otsu, 1979] (Fig. 6.3c-2). Finally, we intersect the foreground pixels of the test image with the binarized mapping values (like an error mask), and, as a result, we know which pixels are properly recognized and which are incorrectly recognized (Fig. 6.3c-3). Therefore, since the background pixels do not provide information, the number of correct pixels is normalized by the foreground pixels.

The time complexity of the algorithm is $O(IJw^2c^2)$, where $I \times J$ are the test image dimensions, w is the warp range parameter, and c is the local gradient context window size. It is important to note that in practice both w and c take low values compared to the image sizes.

Recognition Error (IMEGE)

The BIDM algorithm computes the number of pixels of a test image that are correctly allocated in another reference image according to the IDM model. The algorithm that we used followed the concepts of precision and recall to compute the Image-based Mathematical Expression Global Error (IMEGE).[b] First, we compute the BIDM value from the test image to the reference (precision p). Second, we compute the same value from the reference image to the test image (recall r). Finally, both values are combined using the harmonic mean $f_1 = 2(p \cdot r)/(p + r)$, and we obtain the final error value. Fig. 6.3 illustrates an example of this process.

Rendering the image of a math expression encoding copes with the problem of representation ambiguity. Moreover, IMEGE provides a normalized value in the range $[0, 100]$ than can be interpreted as a visual error (as human beings do) and is not as pessimistic as expression recognition rate. On the other hand, IMEGE can not distinguish the source of the errors although it can identify the misrecognized zones of the math expression. As a visual error, misrecognitions involving larger symbols would affect more pixels than errors produced by smaller symbols. Given that this measure takes the global recognition information into account, it can be very helpful to complement the expression recognition rate and symbol related metrics in order to assess the performance of a system.

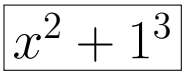
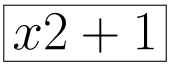
[b] Software available, see Section 8.1.4.

a) Mathematical expression recognition result

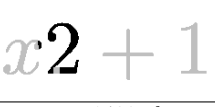
ground-truth = {x^2 + 1^3}

recognition = {x2 + 1}

b) Image generation from ground-truth and recognition

$\text{img}_1 =$  $\text{img}_2 =$ 

c) BIDM computation in both directions

	$\text{img}_2 \rightarrow \text{img}_1$	$\text{img}_1 \rightarrow \text{img}_2$
1		
2		
3		
4	precision = $\frac{1489 \text{ ok}}{2197 \text{ fg}} = 0.6777$	recall = $\frac{1429 \text{ ok}}{2338 \text{ fg}} = 0.6112$

d) Recognition global error

$f_1(\text{precision}, \text{recall}) = 0.6427$

$\text{error} = 100(1 - 0.6427) = \mathbf{35.73}$

Figure 6.3: Example of the procedure for computing the IMEGE measure given a math expression recognition and its ground-truth in LaTeX.

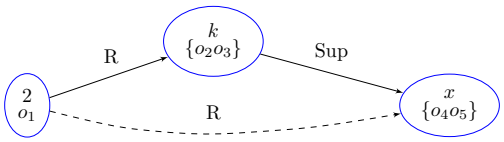
6.2.4 Label Graphs

Zanibbi et al. [2011] proposed a set of performance metrics for online handwritten mathematical expressions based on representing the expressions as *label graphs*. A label graph is a directed graph over primitives represented using adjacency matrices. In a label graph, nodes represent primitives, while edges define primitive segmentation (*merge* relationships) and object relationships. Given a math expression, a label graph is constructed from a symbol layout tree (see Fig. 1.8) such that the strokes in a symbol are split into separate nodes. Each stroke keeps the spatial relationship of its associated symbol, and the nodes inherit the spatial relationships of their ancestors in the layout tree.

Fig. 6.4 shows an example of online handwritten math expression and two label graphs: a label graph for its ground-truth, and a label graph for a recognition result containing errors. Each label graph is displayed such that the dashed edges show the inherited relationships. The adjacency matrix representation is also provided, where the diagonal of the matrix represents the symbol class of each stroke and other cells provide primitive pair labels. These pairs encode the spatial relationships (right, superscript, etc.), where underscore (_) identifies unlabeled strokes or no-relationship, and an asterisk (∗) represents two strokes in the same symbol [Zanibbi et al., 2013].

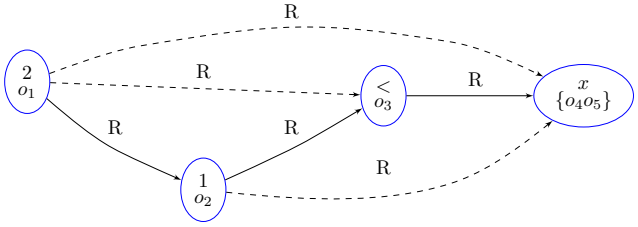
Since the label graph representation contains the information of a mathematical expression at all levels (symbols, segmentation and structure), several metrics can be computed. Given a math expression composed of n strokes, its

Recognition: $2k^x$



2	R	R	R	R
_	***k***	∗	S	S
_	∗	***k***	S	S
_	_	_	***x***	∗
_	_	_	∗	***x***

Ground-truth: $21 < x$



2	R	R	R	R
_	**1**	R	R	R
_	_	<	R	R
_	_	_	***x***	∗
_	_	_	∗	***x***

Figure 6.4: Example of label graph representation of an online handwritten math expression recognition and its ground-truth. The dashed edges are inherited relationships.

ground-truth label graph, and the label graph of a recognition result, Zanibbi et al. [2011] defined the following set of metrics. First, metrics for specific errors:

- Classification error (ΔC): the number of strokes that have different symbol classes (elements of the diagonal of the adjacency matrix) in the label graphs.
- Layout error (ΔL): the number of disagreeing edge labels in the label graphs (off-diagonal elements of the adjacency matrix). This error can be decomposed as the sum of segmentation error (ΔS) and relationships error (ΔR), depending on the type of label of the edges.

Second, metrics at expression level that provide an overall error for a recognition result:

- ΔB_n: the number of disagreeing stroke labels and relationships between two graphs, i.e. the Hamming distance between the matrices of both label graphs. This metric can be computed as

$$\Delta B_n = \frac{\Delta C + \Delta L}{n^2}$$

  This metric will produce more distance for layout errors ($n(n-1)$ elements) than for classification errors (n elements) because is not weighted. For this reason, the next metric was also proposed.

- ΔE: the average per-stroke classification, segmentation and layout errors, such that the three types of errors are weighted more equally. It is calculated as

$$\Delta E = \frac{1}{3}\left(\frac{\Delta C}{n} + \sqrt{\frac{\Delta S}{n(n-1)}} + \sqrt{\frac{\Delta L}{n(n-1)}}\right)$$

In the recognition example of Fig. 6.4, we can see that the symbols 1 and $<$ have been incorrectly grouped as a letter k, and the relationship with the letter x has been incorrectly detected as superscript. The error metrics previously described for this example are:

- $\Delta C = 2$; $\{\boldsymbol{k} \rightarrow \mathbf{1},\ \boldsymbol{k} \rightarrow <\}$
- $\Delta S = 2$; $\{* \rightarrow \_\ ,\ * \rightarrow \mathrm{R}\}$
- $\Delta R = 4$; $\{\mathrm{S} \rightarrow \mathrm{R},\ \mathrm{S} \rightarrow \mathrm{R},\ \mathrm{S} \rightarrow \mathrm{R},\ \mathrm{S} \rightarrow \mathrm{R}\}$

- $\Delta L = \Delta S + \Delta R = 6$
- $\Delta B_n = \frac{2+6}{5^2} = \frac{8}{25} = 0.32$
- $\Delta E = \frac{1}{3}\left(\frac{2}{5} + \sqrt{\frac{2}{20}} + \sqrt{\frac{6}{20}}\right) = 0.4213$

All these label graph Hamming distances are proper metrics [Zanibbi et al., 2011], and the time complexity for the expression level metrics is $O(n^2)$, although in practice only the labeled edges have to be compared, which is must faster for sparse graphs. Furthermore, label graph metrics precisely determine the different types of errors at all levels, which is very useful information. Also, the representation ambiguity of formats like LaTeX or MathML is tackled by the label graph representation and the inheritance of relationships.

Finally, precision and recall at object level (symbols) can also be computed from this representation [Zanibbi et al., 2013]. The last editions of the CROHME competition use these metrics for assessing the performance of the systems [Mouchère et al., 2013, 2014], thus these metrics are becoming the current standard of the state-of-the-art literature in mathematical expression recognition, thanks to the authors that released a great set of open-source tools for computing them[c].

It should be noted that this set of metrics is based on strokes, i.e. for online handwritten mathematical expressions. However, the authors pointed out that they can be applied to images using pixels or connected components as primitives, as well as to other related problems like chemical diagrams, flowchart recognition or tables [Zanibbi et al., 2011, 2013]. In order to do so, the ground-truth has to be provided at primitive level. In the CROHME competitions, the InkML format contains all the required information to build the label graphs, but only LaTeX or MathML annotation would not be enough.

6.2.5 String Matching

One of the most common format for representing mathematical expressions is LaTeX. Recently, Pavan Kumar et al. [2014] proposed a structure encoded string representation computed from LaTeX, although authors comment that it can be extracted from MathML and any other format. They address the problem of performance evaluation of mathematical expression recognition as computing the Levenshtein edit distance between two strings.

Mathematical expressions are two-dimensional structures, while strings are an one-dimensional format. The proposed string encoding is based on consid-

[c]Label Graph Evaluation Library at http://www.cs.rit.edu/~dprl/Software.html

ering symbols left-to-right and two regions for each mathematical symbol, such that *top-left*, *above* and *top-right* spatial relationships are encoded in *northern* region, and *bottom-left*, *below* and *bottom-right* spatial relationships are encoded in *southern* region. This string encoding deals with representation ambiguity by always processing symbols left-to-right and northern region before than southern region.

Pavan Kumar et al. [2014] also defined additional symbols to encode the math expressions as strings, like an *empty* symbol in order to handle special cases. A northern region is delimited by start (N_s) and end (N_e) marks, and a southern region has equivalent start (S_s) and end (S_e) marks. For example, any of the LaTeX strings of Fig. 6.1 encoding the math expression $x_a^2 + 1$ would be represented by the string: $x, N_s, 2, N_e, S_s, a, S_e, +, 1$.

This metric follows the same methodology than EMERS with some differences. The string representation makes the edit distance easier, such that time complexity is $O(n^2)$ while EMERS was $O(n^4)$. Also, string representation does not have the problem of considering symbol types as errors like EMERS in MathML. However, computing the string format requires handling several cases and it is not as straightforward as using a MathML tree. Like EMERS, the string matching also reports an edit distance that is not normalized, such that if two strings are identical their distance is zero. The source of the errors (classification, segmentation or structure) is not detected, and the distance could not be symmetric if the error is weighted taking into account the concept of *levels*.

It seems a good alternative to EMERS, handling better the representation ambiguity and having less time complexity. Although label graph metrics are more complete, it could be useful when only the LaTeX or MathML is provided as ground-truth. Unfortunately, at this moment the software for computing this metrics is not publicly available.

6.3 Summary

In this chapter we first presented and discussed different issues that arise when computing automatic performance evaluation of math expression recognition. There are errors at several levels, ambiguities in representation, and other problems that make comparison of systems difficult.

Afterwards, we described several approaches that have been proposed in order to assess mathematical expression recognition systems. EMERS and string matching are metrics based on computing edit distances between trees and strings, respectively. We also proposed IMEGE a global error calculated

over the image representation of math expressions. These three metrics provide performance metrics at expression level, but the source of the errors (classification, segmentation or structure) is not identified. The benefits and drawbacks of these metrics that have been properly discussed, but they can be helpful to complement errors at symbol level, especially when ground-truth is only provided as a LaTeX or MathML expression.

The set of metrics based on label graphs is the most complete. The proposed representation deals with representation ambiguity and provides information at all levels. Label graphs are based on primitives, so it requires a detailed ground-truth information to construct the graphs (e.g. InkML format). The metrics are well-defined and are becoming the standard reported metrics of last publications in the field, like the last editions of the CROHME international competitions [Mouchère et al., 2013, 2014].

Chapter

7 Experimentation

A complete description of an approach for recognizing mathematical expressions has been provided in this thesis. This pattern recognition problem has to deal with tasks at several levels. Some of the previous chapters focused on solving specific problems within mathematical expression recognition, proposing several features and classifiers. In this chapter, we report the experimentation carried out in order to assess the performance of the different proposals, as well as the performance of the approach at expression level.

This thesis is devoted to develop a method for recognizing mathematical notation. We have described a statistical framework and its associated parsing algorithm for recognizing any type of expression: online, offline, printed or handwritten. Therefore, in this chapter we report several experiments in order to evaluate our proposal.

The datasets of printed and handwritten mathematical expressions used to perform the experiments are described in Section 7.1. Symbol classification is evaluated in Section 7.2 for handwritten symbols, and printed symbol classification is evaluated in Section 7.3. Then, experiments on spatial relationships classification are reported in Section 7.4. Finally, the performance of the approach developed in this thesis for handwritten math expression recognition is assessed in Section 7.5 and its application to printed math expressions is analyzed in Section 7.6.

Chapter Outline

7.1 Datasets

In order to assess and compare the performance of different approaches and methodologies, public datasets are required. Although there was a lack of public resources in the mathematical expression recognition field, fortunately, different datasets have been released during the last years [Suzuki et al., 2005; MacLean et al., 2011; Mouchère et al., 2013]. Below we provide the description of the different datasets we used for validating the developments discussed in this thesis. First we present the datasets of printed mathematical expressions and, second, we describe the datasets of handwritten mathematical expressions.

7.1.1 Printed Mathematical Expressions

Public and large resources of annotated printed mathematical expressions have been available for more years than for handwritten mathematical notation. Below we describe the two datasets of printed mathematical expressions used in this thesis for experimentation.

UW-III: University of Washington

The UW-III database [Phillips, 1998] is a set of document images from different fields that includes 25 journal document pages containing mathematical formulae. Some of the images come from blurred photocopies. Each image has annotated the zones where the math expressions are located, but the mathematical symbols are not isolated. The zones that are annotated are not embedded in the text. We manually isolated and classified the symbols. As a result, the complete database had $2,233$ mathematical symbols belonging to 120 classes. However, many symbol classes had very few samples, such that only 84 classes had at least 4 samples. Some samples of expressions from the UW-III database are shown in Fig. 7.1. It can be noted that expressions are a bit degraded, with problems like noise or touching symbols.

INFTY dataset

Suzuki et al. [2005] released the InftyCDB-1 database, a great resource that contains 467 page images of 30 English articles on pure mathematics. All the characters and symbols are included in the database with their ground-truth information. Therefore, it can be used both for printed text recognition and printed mathematical expression recognition. Other elements like matrices,

$$I(\mathbf{Q}) = \left(b_p - b_m\right)^2 N_p \left| \int_{V_p} e^{i\mathbf{Q}\cdot\mathbf{r}} d^3r \right|^2, \qquad \frac{\partial^2 h}{\partial x^2} = \frac{1}{\alpha} \frac{\partial h}{\partial t}$$

$$\mathrm{Wa}^*(x) = \frac{1}{\left.\dfrac{di}{d\eta}\right|_x \rho_m \mathrm{L}^2 \mathrm{A}_S}. \qquad f_{sh} = \frac{N_{\min} \cdot |\mathbf{U}| \pi d_{in}}{4f\lambda}, \qquad \mathrm{R}_m = \frac{\rho_m \mathrm{L}}{\pi r_c^2}.$$

$$\mathrm{E}[\mu_1(\hat{\mathrm{L}})]^2 = \int [\mu_1(\ell \cap \mathrm{D})]^2 \, d\upsilon(\ell) = \sum_{i,j=1}^{m} \int \mu_1(\ell \cap \mathrm{D}_i) \, \mu_1(\ell \cap \mathrm{D}_j) d\upsilon(\ell)$$

$$= \sum_{i,j=1}^{m} \iint \mathrm{P}[\omega : x \in \mathrm{L}(\omega) \cap \mathrm{D}_i, \; y \in \mathrm{L}(\omega) \cap \mathrm{D}_j] d\mu_1(x) \, d\mu_1(y).$$

Figure 7.1: Examples of printed mathematical expressions from the UW-III dataset.

tables and figures are not labeled in the ground-truth. In this thesis, we are focused on the mathematical information of the database.

The pages of the articles can contain isolated mathematical expressions and also expressions embedded inline the text. The samples are annotated with many useful information both at expression level and at symbol level. Each mathematical expression is located in a page image by the coordinates of its bounding box, and the transcription in MathML, LaTeX and IML is also provided. At symbol level, each symbol is also annotated with its bounding box coordinates in the page image along with the symbol class. Furthermore, there is also information about the type of symbol (operator, Greek, numeric, Roman, etc.), the font (Italic, Bold, etc.) or the quality of the representation (normal, separated or touching symbols, etc.). Finally, each mathematical symbol has also specified the spatial relationship relative to the preceding symbol, which is used to describe the structure of the math expression as a tree.

The number of mathematical expressions in the database is 21K which in turn contain 157K mathematical symbols belonging to 212 classes. Fig. 7.2 shows different examples of mathematical expressions from the Infty dataset. First samples are clean images of complex expressions containing several sym-

bols, operators and relations. Some of the problems related to offline mathematical expression recognition can be seen in these examples. The radical sign of the second expression is separated into two connected components, whereas the T and right parethesis of the bottom-left expression are touching symbols. Moreover, some of the expressions of the Infty dataset are incomplete (see bottom-right expression), which might be a problem for automatic recognition if a system expects complete inputs.

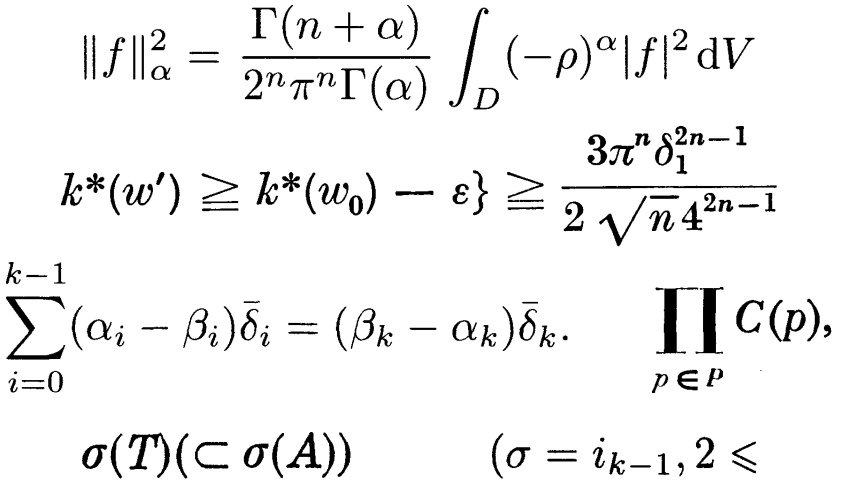


Figure 7.2: Examples of printed mathematical expressions from the INFTY dataset.

7.1.2 Handwritten Mathematical Expressions

Although there was a large public dataset for printed expressions since Suzuki et al. [2005] released the Infty database, similar resources of online handwritten mathematical expressions were released quite recently. In this section we detail the datasets of online handwritten mathematical expressions used in this thesis for experimentation.

MathBrush Dataset

The MathBrush database released by MacLean et al. [2011] represents a great resource for handwritten mathematical expression recognition. It contains 4,654 annotated mathematical expressions written by 20 different writers.

The number of math symbols is 26K and they are distributed in 100 different classes. Each expression is annotated at stroke level with symbol segmentation and symbol identification. The relationships between sets of strokes, corresponding to either symbols or subexpressions, are also provided in SCG format, as well as a full encoding of the mathematical expression in LaTeX, tree format and others.[a] Fig. 7.3 shows some examples of the expressions in the MathBrush dataset. We can see the great writing style variability in the expressions of the first two rows. For instance, a very common symbol like the letter x is written in three quite different ways. Other challenges are strokes that represent more than one symbol (first two letters of trigonometric functions), cursive handwriting (bottom-left expressions), nested expressions (square roots) or the very similar shape of some symbols (number 2 and letter z in bottom-right expression: $\sqrt{\frac{2z^3}{\sqrt{\frac{3z^2}{\sqrt{4z}}}}}$).

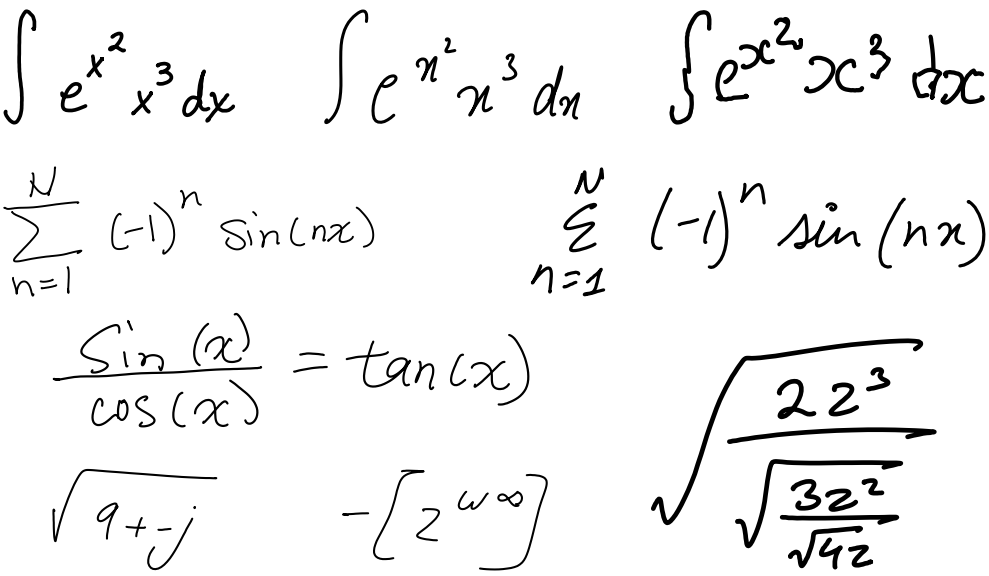


Figure 7.3: Examples of handwritten mathematical expressions from the MathBrush dataset.

[a] Details about formats at https://www.scg.uwaterloo.ca/mathbrush/corpus.shtml

CROHME Datasets

The Competition on Recognition of Online Handwritten Mathematical Expressions (CROHME) has been held annualy since its first edition in [Mouchère et al., 2011]. Initially, the training had 921 handwritten math expressions and was extended to 1,341 on the second edition [Mouchère et al., 2012]. The third edition of the CROHME competition [Mouchère et al., 2013] released a large resource for mathematical expression recognition as a result of combining and normalizing several datasets: the MathBrush database [MacLean et al., 2011], the HAMEX database [Quiniou et al., 2011], the MfrDB database [Stria et al., 2012], the ExpressMatch dataset [Aguilar and Hirata, 2012] and the KAIST dataset. The last edition, to this date, of CROHME [Mouchère et al., 2014] used the same data than the previous competition but provided a different test set. They also included an additional task for recognition of matrices.

In this thesis, we will focus on the dataset of the two last editions of the competition [Mouchère et al., 2013, 2014] since they share the same training data. The samples in the CROHME 2013 dataset are encoded in InkML,[b] a format that includes information of a math expression at all levels: the strokes, the symbol segmentation, and the structure represented in MathML. It contains 8,835 online handwritten mathematical expressions for training and 671 math expressions for testing. These expressions contain about 86K math symbols for training and 6K symbols for testing distributed in 101 classes. The test set of the CROHME 2014 dataset had 986 expressions containing 10K math symbols. The test set of the competitions were collected by the organizing research labs: the IRCCyN/IVC (Université de Nantes, France), the DPRL (Rochester Institute of Technology, USA) and the CVPR (Indian Statistical Institute, India).

The combination of all these datasets from different countries results in mathematical expressions written by many different writers using different styles. Fig. 7.4 shows some samples of mathematical expressions from the databases that compose the competition dataset. It should be noted that the training set of the CROHME competition includes the MathBrush database previously described.

Finally, CROHME 2014 introduced a novel task on recognizing handwritten math expressions containing matrices, which require specific treatment due to more complex spatial relations and dimension restrictions. The organizers released 362 expressions for training and the test set comprised 175 expressions. Examples of this type of formulae are shown in Fig. 7.5.

[b] http://www.w3.org/TR/InkML/

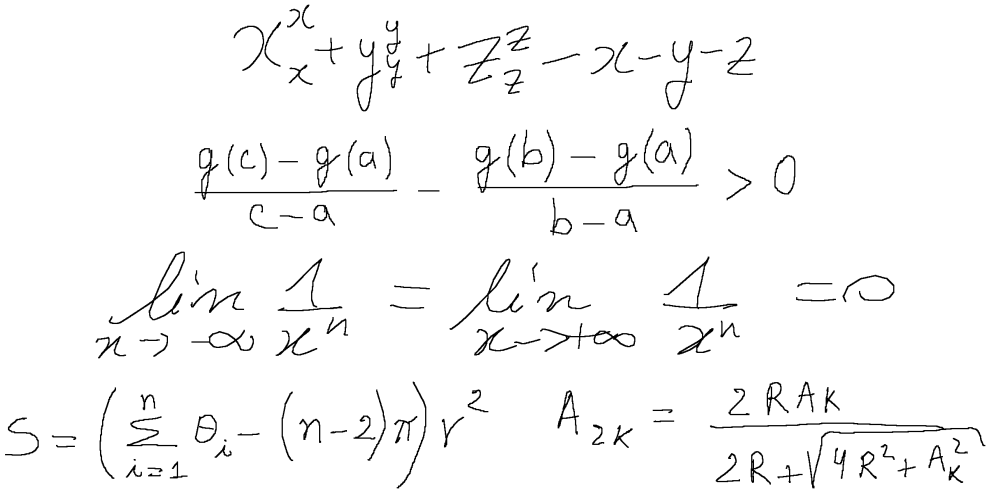


Figure 7.4: Examples of online handwritten mathematical expressions from the CROHME 2013 competition dataset.

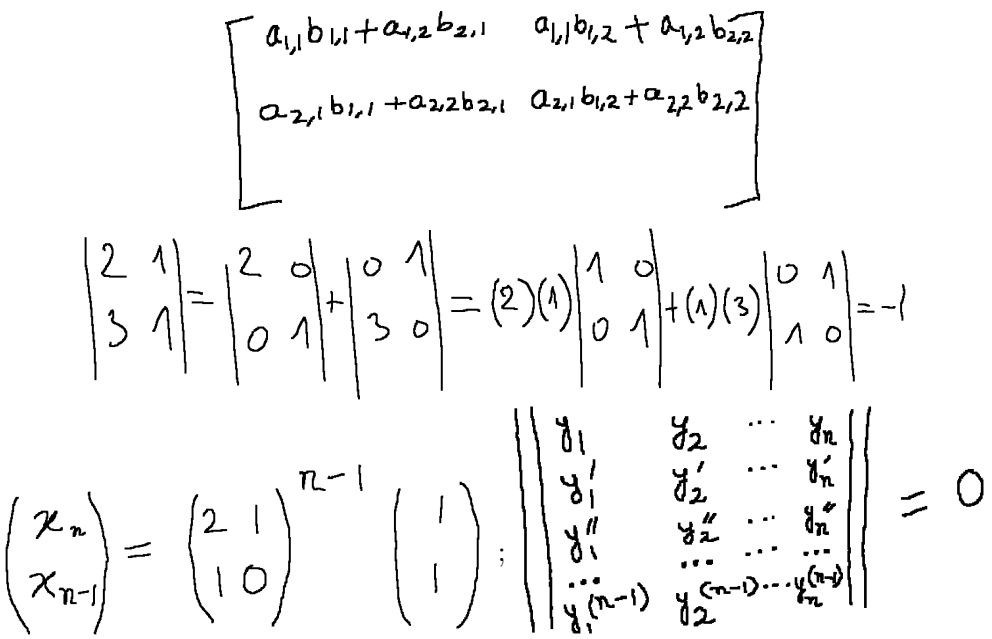


Figure 7.5: Examples of mathematical expressions for the matrix recognition task of the CROHME 2014 competition.

7.2 Classification of Handwritten Symbols

In Section 3.3, dedicated to handwritten symbol classification, we described several sets of features and different classifiers. We carried out several experiments in order to assess the performance of the proposed features and classifiers in this task.

In the following sections we report three main evaluations. First, a comparison between HMM and RNN classifiers. Second, we assess the performance of both online and offline features for symbol recognition, as well as their combination. Finally, we compare our symbol classification methodology to other proposed techniques.

7.2.1 Classifiers Evaluation: HMM and RNN

Two classifiers for sequence classification have been proposed for recognizing handwritten mathematical symbols: HMMs and RNNs. We carried out some experiments in order to compare the performance of both classifiers. To this end, we also tested the recognition rate achieved using online features and the hybrid extension, because context information is an important factor in this evaluation.

For this experimentation we used the MathBrush dataset (Section 7.1.2). Since it was released, some approaches have reported results using this dataset. Therefore, in addition to comparing the performance of HMMs and RNNs, we carried out the same experimentation described by Hu and Zanibbi [2011] in order to obtain comparable results and validate our experiments. Thus, we discarded six symbol classes ($\leq, \neq, <, \lambda, \Omega$ and 'comma') because they had less than 50 samples in the train set, and the symbol 'dot'. As a result, 93 different mathematical symbols were considered. For each class, we used 90% samples as the train set (22,305) and the remaining 10% as the test set (2,531). Finally, we carried out 20 trials where the train and test sets were chosen randomly, and symbol recognition results are stated as the mean and the standard deviation computed across the trials.

Below we detail the experiments carried out with HMMs and RNNs for mathematical symbol recognition. Results are presented and analyzed for each classifier, and at the end of this section we discuss the differences in performance of both classifiers.

Hidden Markov Models experimentation

Mathematical symbol classification was performed using HMM with different number of Gaussian mixtures per state (Section 3.3.4). We used left-to-right HMM with variable number of states s_c per class c, computed as $s_c = l_c/k$, where l_c is the average number of feature vectors per class in the train set, and k is a design parameter that measures the average number of feature vectors modelled per state (*state load*). This parameter was set to $k = 6$ and s_c forced to be in range $[3, 15]$ through preliminary experimentation (or using previous knowledge). The training process of the HMM was done by means of the Baum-Welch algorithm [Baum, 1972]. This is a generative training procedure that does not require a validation set.

The extension of the online features with offline information had several parameters to be tuned. These parameters were tuned using one of the 20 trials, then the best values were set for the remaining experiments. First, each sample was rendered to an image of fixed height h as described in Section 3.3.2. Then, different context-window sizes w were tested, and for each size several number of PCA dimensions d were also evaluated.

Figure 7.6 shows the results for hybrid features when tuning the parameters. Image height $h = 20$ obtained the best results, because it provided a better representation than smaller images and it also included more information on each pixel than larger images (Figure 7.6 left). Regarding context-window size (w) and PCA dimensions (d), the best results were provided by $w = 15$ for both $d = \{8, 14\}$ (Figure 7.6 right), but smaller values were preferred to obtain less complex models. Consequently, we finally selected the values $h = 20$, $w = 15$ and $d = 8$ because they obtained the best results adding only 8 new offline features per frame.

Figure 7.7 shows the HMM mathematical symbol classification results in the trial used for tuning the parameters for both sets of features and different number of Gaussian mixtures per state. Results showed that the hybrid features greatly improved the symbol recognition rate with respect to online features. Using 128 Gaussian mixtures and online features the average recognition rate was 82.3% whereas hybrid features raised the recognition accuracy up to 88.2%.

Recurrent Neural Networks experimentation

In order to make a fair comparison, we carried out several experiments with RNN analogously to HMM experimentation. We used exactly the same train and test sets and the same feature vectors.

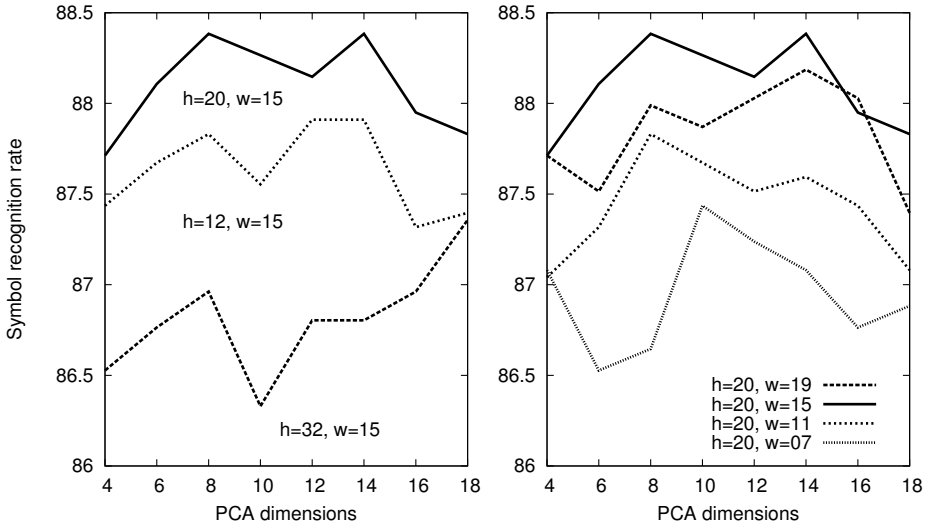


Figure 7.6: Hybrid features parameter tuning using HMM for different image heights (h), context window sizes (w) and PCA dimensions. Symbol recognition rate for one of the 20 trials.

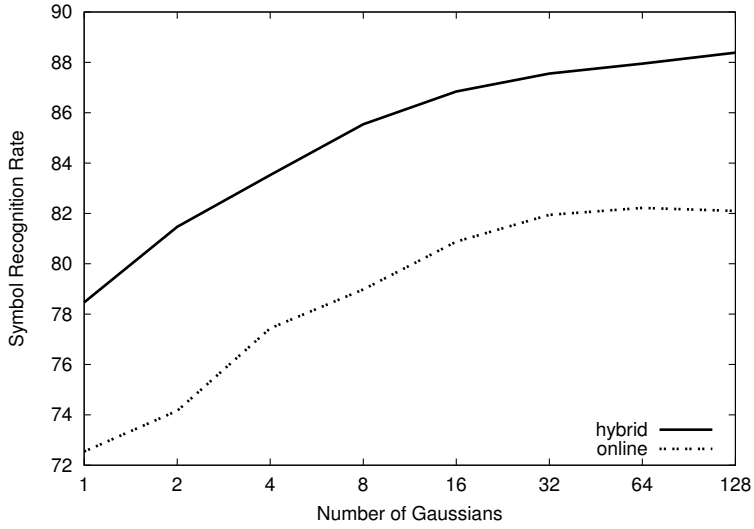


Figure 7.7: HMM symbol recognition rate for different number of Gaussian mixtures per state using both online features and hybrid features. Results computed for one of the 20 trials.

The RNN had a BLSTM architecture (Section 3.3.4). The size of the input layer was determined by the feature set: 7 for online features and 15 for hybrid features. The output layer size was 93, the number of mathematical symbol classes. Finally, the size of the forward and backward hidden layers was tuned for one of the 20 trials and then that value was fixed for the remaining experiments (Figure 7.8). In the final RNN configuration for both online features and hybrid features, the forward and backward hidden layers each contained 70 LSTM memory blocks.

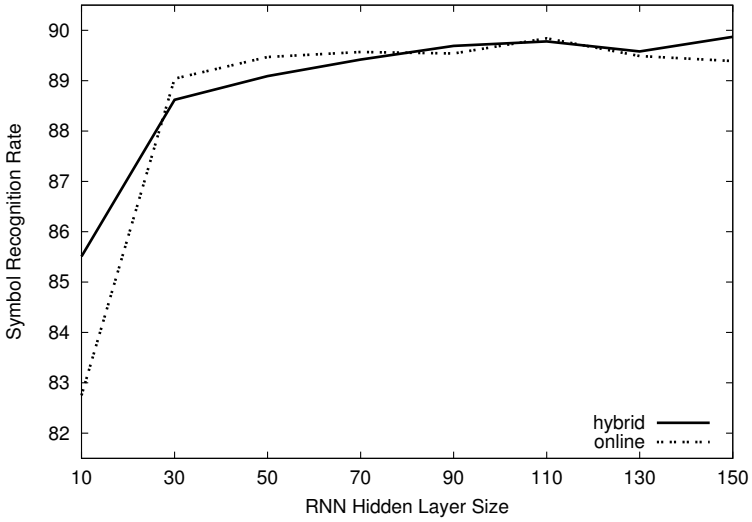


Figure 7.8: RNN symbol recognition rate in one of the 20 trials for different hidden layers size using both online features and hybrid features.

Following [Graves et al., 2009], the network weights were initialized with a Gaussian distribution of mean zero and standard deviation 0.1. The network was trained using online gradient descent with a learning rate of 0.0001 and a momentum of 0.9. In order to use the same train and test set that on HMM experimentation, we extracted a validation set from the train set using the same criterion of 90%/10% samples per class. Then, the error rate was recorded every 5 epochs on the validation set and training was stopped when performance had ceased to improve on the validation set for 50 epochs. Because of the random initialization, each experiment was repeated four times and we calculated the average number of epochs to obtain the best network according to the validation set. Finally, we trained the RNN during that number

of epochs using the original train set.

We carried out 20 mathematical symbol classification experiments using RNN as previously described, where each trial included four repetitions and the average recognition rate was considered. Final results for RNN and HMM classifiers using both online features and hybrid features are shown in Table 7.1, where both top-1 and top-5 recognition rates are reported. The symbol recognition rate achieved using online features by RNN clearly outperformed HMM. The RNN results obtained with online features were even slightly better than the HMM results with hybrid features. However, the hybrid features in combination with RNN did not produce the great improvement observed in HMM.

Table 7.1: Classification results with HMM and RNN classifiers and two sets of features: online and hybrid.

Classifier	Features	Top-1	Top-5
HMM	online	82.3 ± 0.5	98.1 ± 0.3
HMM	hybrid	88.2 ± 0.5	99.2 ± 0.2
RNN	online	89.3 ± 0.4	99.2 ± 0.2
RNN	hybrid	89.4 ± 0.3	99.3 ± 0.2

Discussion

We followed the experimentation described in Hu and Zanibbi [2011] over a large public database. In that work the best recognition rate among 20 trials was 82.9% using HMM and four online features. Thus, our results are comparable since we obtained a 82.3% average recognition rate with HMM and seven online features. Additionally, we tested two sets of features and two different classifiers.

When using the online features, RNN outperformed HMM increasing the symbol recognition rate from 82.3% up to 89.3%. This great improvement comes mainly for two reasons. First, the RNN classifier learns using discriminative training, whereas HMM learns through generative training. Discriminative training usually obtains better results in classification tasks. Moreover, RNN take into account context information which is very important for handwriting recognition, whereas HMM assume that frames are independently observed.

Regarding hybrid features, they provided a great improvement in HMM compared to using only online features, but recognition results barely differed in RNN. In our opinion, HMM obtained such good results because the proposed offline features included the context information on each point, such that they helped to solve the lack of HMM regarding contextual modelling. For that reason, despite hybrid features also added offline information of the sample, RNN did not present differences with respect to the results obtained with online features. The differences in performance between RNN and HMM corroborate the findings from Graves et al. [2009].

In conclusion, RNN outperforms HMM in this handwritten math symbol classification task. RNN are faster than HMM in classification, and using only 7 online features the RNN obtained top-1 recognition rate of 89.3% whereas HMM obtained 82.3%. The hybrid features increased the HMM performance up to 88.2% top-1 recognition rate, but still results were worse than those of RNN. Furthermore, hybrid features required additional computation, and 15 features per frame results in more complex models. Contextual modelling was the key factor provided by both hybrid features and RNN, as well as the discriminative training used in this classifier.

7.2.2 Features Evaluation: Online and Offline

In this section we detail the experimentation carried out to assess the performance of the different online and offline features proposed for handwritten mathematical symbol classification. The symbol classifiers were BLSTM-RNN because yield better results than HMM. We used the RNNLIB library [Graves, 2010] and the software for computing the different features are also available (see Section 8.1.3).

Experimental setup

We used the dataset of the CROHME 2013 competition (see Section 7.1.2) for the experimentation. Although this dataset only contains online handwritten mathematical expressions, we can render the offline version of each symbol as described in the hybrid features definition (see Section 3.3.2). In this case the image height was set to $H = 40$ pixels.

In adition to the train and test sets of the CROHME 2013 competition, we needed a validation set for tuning parameters of some features and also for training the RNN classifier. Hence, we first extracted a validation set by taking 10% of the samples of each class in the original train set, and the

remaining 90% were used for training. This way we also kept proportionally the distribution of the classes.

We used the same configuration of the RNN classifier for every experiment. The size of the input layer was determined by the feature set and the output layer size was 101, i.e. the number of symbol classes. The forward and backward hidden layers each contained 100 LSTM memory blocks. The network weights were initialized with a Gaussian distribution of mean zero and standard deviation 0.1. The network was trained using online gradient descent with a learning rate of 0.0001 and a momentum of 0.9. This configuration has obtained good results in handwritten text recognition [Graves et al., 2009] and in previous experiments comparing HMMs and RNNs.

The experimentation was carried out in two steps. First, we trained the network until the error had ceased to improve on the validation set for 50 epochs. As the RNN results depend on the random initialization, we performed four experiments with the validation set in order to compute the average number of epochs $\bar{e}$ required to obtain the best network. Then, we trained four RNNs using the full training set during $\bar{e}$ epochs. Finally, we classified the test set with the RNNs trained with the full training set and the average recognition rate is reported along with the standard deviation computed across the four experiments.

Results

Following we report the classification performance of every set of features while recognizing the CROHME 2013 test set using RNN classifiers. The online features do not need any parameter to be tuned, so we carried out the experiment as previously defined. Regarding offline features, the PRHLT features computed 60-dimensional vectors for each column of the rendered images and the FKI features extracted 9 features per column. On the other hand, RWTH and POLAR features had some parameters to be tuned. We tried several sizes w of the sliding window in RWTH features. The polar features radius was set to half the height of the image, in this case $r = 20$, and we tested different values of circles (n) and arcs (m). We also tried different number of dimensions for the PCA projection.

The parameters of the features in the final experiment were obtained according to the error obtained with the validation set. The RWTH features were extracted using a sliding window of width $w = 5$ pixels and projected to 30 dimensions, whereas the POLAR features were computed with $n = 5$ circles, $m = 12$ arcs and also projected to 30 dimensions. This configuration was also used with the vertical repositioning method of both sets of features.

Finally, given that we have computed both online and offline features, the next natural step was to combine both classification results in order to obtain better performance [Keshari and Watt, 2007]. We combined the posterior probability of an RNN with online features and the posterior probability of an RNN with offline features using linear interpolation and a weight parameter (α).

$$p(s \mid b_i) = \alpha \cdot p_{\text{RNN}}(s \mid \mathcal{F}_{\text{on}}(b_i)) + (1 - \alpha) \cdot p_{\text{RNN}}(s \mid \mathcal{F}_{\text{off}}(b_i)) \tag{7.2.1}$$

where the probability for one model is computed as defined in Eq. (3.3.2). We set $\alpha = 0.5$ for every combination.

The results are shown in Table 7.2 such that there are three sets of results: using online features, using offline features and their combination. We report top-1, top-2 and top-5 recognition rates. Furthermore, there are several classes that produce many classification errors because they have very similar shape but different semantic [Hu and Zanibbi, 2013]. Therefore, we also computed top-1 recognition rate where those similar classes were merged (TOP_1'). The symbols merged in this error were: $\{1, |, \prime, \text{comma}\}$, $\{P, p\}$, $\{S, s\}$, $\{C, c\}$, $\{X, x, \times\}$, $\{V, v\}$ and $\{o, 0\}$. It should be noted that we are dealing with the handwritten version of these symbols.

Discussion

The experimentation summarized in Table 7.2 produced several interesting results. First, although online data normally yields better results than offline data [Plamondon and Srihari, 2000], we obtained higher recognition rate values with offline features than using online features. Results show that online features obtained on test set a 82.5 top-1 recognition rate whereas PRHLT, FKI and RWTH offline features improved up to 83.9, 84.1 and 83.4 respectively. In our opinion, an important factor for these results could be that the CROHME database comprises samples from many different writers and datasets. This can cause that the online samples were more sensitive to the writing style, whereas the image representation of the symbols could present less variance.

Regarding the performance of the offline features, there are several results to take into account. First, we can see that PRHLT, FKI and RWTH features obtained the best results outperforming online features. On the other hand, the proposed POLAR features provided a good performance although not as good as the rest of offline features, even slightly worse than the online results. It should be noted that this was an initial test of polar features for this task

Table 7.2: Classification results with RNN classifiers and several sets of features: online, offline and their combination.

Features	TOP_1	TOP_2	TOP_5	TOP'_1
Online	82.5 ± 0.3	92.3 ± 0.1	96.8 ± 0.1	88.3 ± 0.2
PRHLT	83.9 ± 0.4	93.4 ± 0.2	97.6 ± 0.1	89.9 ± 0.4
FKI	84.1 ± 0.2	93.6 ± 0.1	97.8 ± 0.1	90.3 ± 0.2
RWTH	83.4 ± 0.3	93.1 ± 0.2	97.5 ± 0.0	89.6 ± 0.3
RWTHvr	82.9 ± 0.2	92.2 ± 0.3	97.4 ± 0.1	88.8 ± 0.2
POLAR	81.2 ± 0.4	91.2 ± 0.3	96.7 ± 0.1	87.2 ± 0.3
POLARvr	80.8 ± 0.3	90.4 ± 0.3	96.4 ± 0.1	86.6 ± 0.4
On. + PRHLT	87.1 ± 0.2	95.0 ± 0.1	98.3 ± 0.1	92.7 ± 0.2
On. + FKI	86.8 ± 0.4	95.3 ± 0.2	98.4 ± 0.1	92.6 ± 0.2
On. + RWTH	86.7 ± 0.3	94.9 ± 0.2	98.4 ± 0.1	92.6 ± 0.1
On. + RWTHvr	86.5 ± 0.2	94.7 ± 0.2	98.4 ± 0.1	92.3 ± 0.0
On. + POLAR	86.1 ± 0.4	94.5 ± 0.2	98.2 ± 0.1	91.9 ± 0.1
On. + POLARvr	86.0 ± 0.2	94.2 ± 0.2	98.1 ± 0.1	91.7 ± 0.1

and better results could be obtained with more advanced configurations of the shape descriptors [Su et al., 2013].

We would like to remark the results obtained by the FKI features because they obtained the best results with only 9 features, while the rest of offline features required 30 or 60 dimensions and more complex calculations. We think that FKI features obtained such a good performance because the RNNs can take advantage of context. Hence, it does not require features to include context in their representation, like RWTH or POLAR, because the model itself can manage it. Another result that may suggest this conclusion is that the recognition rate of RWTH features in validation set worsened with wider sliding windows.

With respect to the vertical repositioning method, this technique has really improved the results in handwritten text recognition [Doetsch et al., 2012]. Nevertheless, in this task results did not present any improvement or they were even slightly worse. Our intuition is that this could be caused due to we are tackling the classification of isolated handwritten math symbols. This means that each sample is already segmented, whereas in handwriting text recognition the segmentation is a complex problem solved implicitly by the

model, like HMMs or RNNs [Graves et al., 2009].

The combination of the classification results of online features and offline features led to significant improvements in the achieved recognition rate. We combined the posterior probabilities of both results according to Eq. (7.2.1) with $\alpha = 0.5$. Every combination between online and offline systems obtained better error rates, with relative improvements up to 20%.

Finally, we could see that there is an important gap between top-1 and top-2 results, and top-5 recognition rates are really high. This behaviour shows that even though there is still room for improvement, the classifier provides very good results. We expect that with integration of this classifier in a mathematical expression recognition system, the contextual information of the expression could help to solve many misclassifications. This can also be seen according to the reported TOP'_1 recognition rate, because the combined classification achieved almost 93% recognition rate if those similar shaped classes are not considered as errors. In addition to the classes detailed in TOP'_1 computation, many errors were caused by very similar classes like: $\{5, s\}$, $\{t, +\}$, $\{comma,)\}$, $\{q, 9\}$ or $\{z, 2\}$.

7.2.3 Comparison with other approaches

As a result of the previous experiments, we concluded that our best mathematical symbol classifier was the combination of an RNN trained with online features and an RNN trained with offline features (FKI). The CROHME 2014 competition included an isolated symbol recognition task. We participated with our approach trained on the training set of the competition (Section 7.1.2) and tuning the parameters using the test set of the previous edition (CROHME 2013). As every edition of CROHME, the contestants submited their systems to the organizers and they classified the hidden test.

The task of symbol recognition had in turn two evaluations: the accuracy of the classifier with respect to the true symbols, and the ability of the classifier to accept true symbols while rejecting junk samples [Mouchère et al., 2014]. For the evaluation without junk class, we trained our classifier as previously described. For the scenario with rejection option, the organizers provided a tool for extracting junk samples from the training expression. The RNN classifier learns the posterior probability of the training set, thus the amount of training samples for each class affects the prior probability of each symbol class. In this experimentation, we fed the RNN classifier with the same number of samples of true symbols and junk samples.

Seven different systems participated in the competition (note that System VII did not participate in this task):

System I - Universitat Politècnica de València
This is the approach proposed in this thesis for online handwritten symbol classification. A combination of two RNN classifiers: one RNN trained with online features (7 per frame) and another RNN trained with offline features (9 per frame).

System II - University of São Paulo
Julca-Aguilar et al. [2014] used a Multi Layer Perceptron (MLP) neural network classifier with a combination of fuzzy shape context features and online features.

System III - MyScript
This approach used a set of features that combines dynamic information from the ink signal (position, direction and curvature), and static information from a bitmap representation of the ink (projections and histograms). The symbol classifier is an MLP neural network.

System IV - Rochester Institute of Technology, DPRL
The classifier proposed by Davila et al. [2014] used SVM with a Gaussian Kernel and several sets of online and offline features: global features, crossings features, fuzzy histograms of points and fuzzy histograms of orientations.

System V - Rochester Institute of Technology, CIS
In this case, the authors used an SVM classifier and five features: pen-up/down, the density in the center of a 3×3 matrix containing symbol strokes, normalized y-coordinate, sine of curvature and cosine of vicinity from slope [Liwicki and Bunke, 2009].

System VI - Tokyo University of Agriculture and Technology
This system normalizes each input symbol pattern with the Line Density Projection Interpolation normalization method. Then a 512-dimensional feature vector is extracted based on Normalization-Cooperated Gradient Feature. The feature vector is reduced to 300 dimensions by FLDA. The reduced vector is classified by a Generalized Learning Vector Quantization classifier. For classifying invalid symbols more accurately, patterns are clustered into 64 clusters by using the LBG algorithm for training.

System VIII - Athena Research and Innovation Center, ILSP
Simistira et al. [2008] developed a system based on a template elastic matching distance such that the mathematical symbols were encoded using pen-direction features and the 8-level Freeman chain coding scheme.

The results of the competition are shown in Table 7.3, where the reported metrics are top-1 recognition rate (TOP_1) and true mean position (TMP). The true mean position is the mean position of the correct label in the n-best output of the classifier. In the evaluation with junk samples, the true acceptance rate (TAR) and false acceptance rate (FAR) are also reported.

Table 7.3: Results of the isolated symbol recognition task of the CROHME 2014 competition. System I is the approach developed in this thesis.

System	101 classes		101 classes + junk class			
	TOP_1	TMP	TOP_1	TMP	TAR	FAR
I	91.24	1.20	no reject option			
	89.79	1.25	84.14	1.28	80.29	6.44
II	82.72	1.60	79.11	1.50	82.49	14.63
III	91.04	1.19	85.54	1.23	87.12	10.39
IV	88.66	1.27	83.61	1.29	83.52	9.03
V	85.00	1.37	71.19	1.48	86.84	36.85
VI	84.31	1.53	no reject option			
	82.08	1.63	76.24	1.57	77.98	16.47
VIII	77.25	1.68	no reject option			

The proposal developed in this thesis (System I) obtained the best results in the evaluation without rejection option (91.24%) and performed second-best in the evaluation considering the junk class (84.14%). In this later scenario, we obtained the lowest false acceptance rate (lower FAR is better) but also a lower true acceptance rate than other systems (higher TAR is better). This is directly related to the percentage of junk samples used for training. We could balance the performance of the RNN classifier regarding this issue by selecting different ratios of junk samples during the training process.

7.2.4 Summary

In this experimentation we have evaluated different options for classifying handwritten mathematical symbols. We first compared the performance of HMMs and BLSTM-RNNs, as well as two sets of features: online features and hybrid features. RNN classifiers outperformed HMM in this symbol recognition task. The hybrid features significantly improved the recognition rate of

HMMs but the accuracy of RNN barely differed. Contextual modelling was the key factor provided by both hybrid features and RNN, as well as the discriminative training procedure of RNNs compared to the generative training of HMMs.

Later, we studied the performance of several sets of offline features for handwritten math symbol classification. We tested different well-known features, a novel set of features based on polar histograms and the vertical repositioning method. Results show that offline features provided better results than online features, and their combination produced up to 20% relative improvement. FKI offline features presented the best performance due to they obtained the best recognition rates using only 9 features. The vertical repositioning method did not improve results in this task.

Finally, we compared our approach based on the combination of an online RNN and an offline RNN to other proposals. We reported the result of a recent international competition [Mouchère et al., 2014] using a public dataset. Our symbol classifier obtained the best performance in several metrics, showing state-of-the-art competitive results.

7.3 Classification of Printed Symbols

In Section 3.4, we described four different classifiers for printed mathematical symbol recognition. Below, we report the experimentation carried out in order to assess the performance of the different proposals.

7.3.1 Experimental setup

We performed experiments on two public datasets of printed mathematical expressions (see Section 7.1). In both datasets, we discarded touching symbols and only considered symbol classes that had at least four samples. First, we used 2,076 mathematical symbols belonging to 84 classes of the UW-III database. We randomly extracted 75% of the samples for training and the remaining 25% samples for testing. As the UW-III is a small dataset, we repeated this process 100 times. Second, we used the INFTY dataset which contains roughly 157K symbols. Given the large size of this dataset, we limited the maximum amount of training and test data to make the experimentation more feasible. We composed four training sets of increasing size (5K, 10K, 20K, 50K) and one test set (5K). These training and test datasets were chosen at random but keeping the actual distribution of symbols of the original dataset. The total number of classes for the experimentation was 183.

Below we detail the configuration of the classification techniques described in Section 3.4.3. For the k-NN classifier, we tested values of $k = \{1, 3, 5\}$ without using any technique of prototype removing. We initially divided the set of prototypes into three categories based on the aspect ratio, thus we greatly reduced the number of comparisons. A test sample was only compared with the set of prototypes of its category. If we did not divide the set of prototypes according to the aspect ratio, the classification error rate remained the same.

The WNN classifier required tuning some parameters during the training process. For this reason, we adjusted them using one of the 100 repetitions of the UW-III experimentation and set those values to the remaining experiments on both datasets. A public implementation of this classifier is available.[c]

For the SVM classification, we used the SVM^{multiclass} tool[d] with a linear kernel, because Malon et al. [2008] reported better classification results with linear kernel for printed math symbol recognition on the INFTY dataset. The parameters were tuned on the UW-III as previously described for the WNN classifier.

[c] http://users.dsic.upv.es/~rparedes/research/CPW
[d] http://svmlight.joachims.org/svm_multiclass.html

Finally, for the HMM classification technique, we tested different number of Gaussian distributions per state using the HTK toolkit.[e] The HMMs were left-to-right models with different number of states for each symbol class, depending on the average width of the image samples of each class. The number of states ranged from 1 (e.g. vertical bar) to 15 (e.g. trigonometric functions or square roots).

7.3.2 Results and Discussion

The experimental results on the UW-III dataset are shown in Table 7.4. We can see that the best results were obtained by the SVM classifier. Regarding the classification error with the k-NN classifier, it increased as k increased because there were not enough "similar" prototypes to choose in classes with few prototypes. We tested this hypothesis by removing the classes that had less than 8 prototypes per class (column ≥ 8 in Table 7.4), and 16 prototypes per class (column ≥ 16 in Table 7.4). Thus, we can see that the difference in performance between 1-NN and 5-NN classifiers decreased from 3.26 with more than 4 samples per class to 1.52 with more than 16 samples per class; which is consistent with our hypothesis.

Table 7.4: Average classification error rate for the UW-III data set.

	Number of samples per class		
Classifier	≥ 4	≥ 8	≥ 16
1-NN	6.34 ± 0.08	5.44 ± 0.08	5.05 ± 0.07
3-NN	8.27 ± 0.09	6.71 ± 0.07	5.70 ± 0.08
5-NN	9.60 ± 0.08	7.78 ± 0.07	6.57 ± 0.09
WNN	5.88 ± 0.07	5.23 ± 0.08	5.02 ± 0.09
SVM	4.85 ± 0.05	4.27 ± 0.04	4.24 ± 0.05
HMM	6.42 ± 0.09	6.36 ± 0.08	5.71 ± 0.07

We also observed that the WNN classification technique was very competitive, slightly improving the performance of the 1-NN classifier. HMMs obtained good results, although worse than other classifiers. Sequential classification might be less convenient for this problem than recognition with global representation because printed isolated math symbols are fairly regular.

[e] http://htk.eng.cam.ac.uk/

On the other hand, Table 7.5 shows the results of the experimentation with the INFTY dataset. In all cases, the results improved as the size of the training set increased. The best result were obtained with SVM, but WNN obtained analogous competitive results. In this case, as more samples were available, the learning process of the WNN classifier produced a greater improvement with respect to the 1-NN classifier.

The k-NN classification rule obtained similar values for different values of k with a large amount of prototypes, given that more samples per class were available in the training set. HMMs obtained competitive results with less samples, but they did not improve as much as the other classifiers when the training set increased.

Table 7.5: Classification error rate for the InftyCDB-1 data set.

	#Training samples			
Classifier	5K	10K	20K	50K
1-NN	6.3	4.5	4.3	3.3
3-NN	7.0	5.0	4.3	3.2
5-NN	7.9	5.5	4.4	3.5
WNN	4.8	3.5	3.4	2.8
SVM	4.5	3.4	3.0	2.6
HMM	4.8	3.9	3.8	3.8

In the four classification techniques, approximately 50% of the errors involved classes *overline*, *minus*, *fractional line*, *underline*, and *hyphen*. These symbols have almost the same shape, and they should be merged and distinguished by structural methods.

A classification error rate of 1.5% was reported on the INFTY database by Suzuki et al. [2003]. However, it should be taken into account that they classified entire printed expressions, thus symbol classification was helped with structural information. Other printed math symbols classification techniques are reported in Garain et al. [2004] on a different database. In that work, first the symbols are confidently classified if some special primitives appear in the mathematical symbol. This allows the system to classify 39% of the symbols, obtaining an error rate of about 1.7%. For the remaining symbols, the best obtained results after combining three classifiers was 6.2% error rate.

7.3.3 Summary

In this experimentation we have compared four classifier for printed mathematical symbol recognition. The k-NN classifier and SVM are well-know models that had been used for this task. We also proposed and tested two additional classifiers that had not been used in this field: WNN and HMMs.

The best results were obtained with SVM and WNN classifiers. Classic k-NN are simple and provided good performance as the number of training samples increased. Finally, HMMs obtained competitive results, although performed worse than the other proposals. We concluded that, although HMMs have been successfully used for handwritten text recognition, classifiers based on global representation are more appropriate for this problem.

7.4 Spatial Relationships Classification

In this section we evaluate the features proposed in Section 4.2 for spatial relationship classification. We report results using the MathBrush dataset of handwritten mathematical expressions (Section 7.1.2). This dataset contains 21,238 spatial relationships distributed in five classes: *right* (BC), *below* ($\frac{B}{C}$), *subscript* (B_C), *superscript* (B^C) and *inside* ($\sqrt{C}$). Each expression has explicitly annotated the spatial relationships between symbols and subexpressions.

We divided the dataset randomly into 10 partitions while keeping the distribution of spatial relations roughly uniform over the partitions. Then we performen 10 experiments such that the training set contained 80% of the samples for each class, and the remaining 20% comprised the test set. We used an SVM classifier with a Gaussian kernel in our experiments. In order to tune the parameters for training the SVM classifier or to select the parameters of the shape-based features, we also divided the training set (80%) into 70% for training and 10% for a validation set. That split also kept the distribution of the classes, and the best parameters in the validation set were used to finally train the SVM with the complete training set. The error was computed using the test set and averaged for each one of the experiments.

7.4.1 Geometric Feature Results

We performed several experiments to test the geometric features previously described in Section 4.2.1. First, we computed the classification error such that the normalization factor F is equal to the height of the parent region B (GEO_1). Then, we classified the spatial relations using another normalization factor, the distance between the center of the bounding boxes (GEO_2). Finally, we normalized the features by the height of the region resulting after combining regions B and C (GEO_3). Moreover, we also extracted the geometric features GEO_2 and GEO_3 without using the information about symbol categories (ascendant, descendant, normal, middle) in order to measure the influence of this decision.

The results in Table 7.6 show that the normalizations by center point distance (GEO_2) and combined height (GEO_3) provided the best results. The normalization factor by the height of the parent region (GEO_1) performed worse, increasing the mean classification error rate from 2.83% to 3.62%. This is because short symbols (e.g. a fraction line) could lead to poor normalization in GEO_1. The results also show that error improves when computing a vertical centroid based on symbol typographic categories.

Table 7.6: MathBrush spatial relationship classification results. For each feature the number of features (#) and whether typographic symbol classes are used (Cat.) are shown.

Feature	#Features	Cat.	% Error ($\mu \pm \sigma$)
GEO_5: F = height(B∪C)	9	No	3.55 ± 0.20
GEO_4: F = dist(centers)	9	No	3.48 ± 0.39
GEO_3: F = height(B∪C)	9	Yes	2.83 ± 0.21
GEO_2: F = dist(centers)	9	Yes	2.84 ± 0.16
GEO_1: F = height(B)	9	Yes	3.62 ± 0.34
SHP: $n = 15, m = 20$	35	No	3.34 ± 0.21
GEO_2 + SHP	44	Yes	2.70 ± 0.29

7.4.2 Shape-Based Feature Results

The polar histogram-based descriptor presented in Section 4.2.2 has parameters that need tuning, specifically the number of circles n and angles m, and the number of principal components d to project. We performed a grid search for several sizes of the descriptor ($n = \{3, 5, 10, 15, 20\}$ and $m = \{8, 12, 16, 20, 24, 28, 32\}$), and for each size, different numbers of principal components were also tested (variance explained from 10% to 90% in increments of 10%). We used one of the 10 partitions extracted from the experimentation to tune these parameters (see Figure 7.9).

We chose $n = 15$, $m = 20$ and $d = 35$ (50% of total variance) as the parameters to perform the experiments for the shape-based geometric features (SHP). For small grid sizes, results were best when high percentage of variance were accounted for in the PCA dimensions (70%-90%). However, as the grid size increased the variance in bin counts also increased, with the best results being obtained when keeping components covering roughly 50% of the variance. The polar histogram features obtained a mean classification error of 3.34% (Table 7.6), without including symbol typographic classes; this is comparable to the accuracy obtained using the geometric features without symbol typographic classes, where best error was 3.48%.

We tried adding to the shape-based descriptor the information about symbols categories in the relation by displacing the centroids G_A or G_B following the methodology described in Section 4.2.1. However, this led to weaker results in this representation.

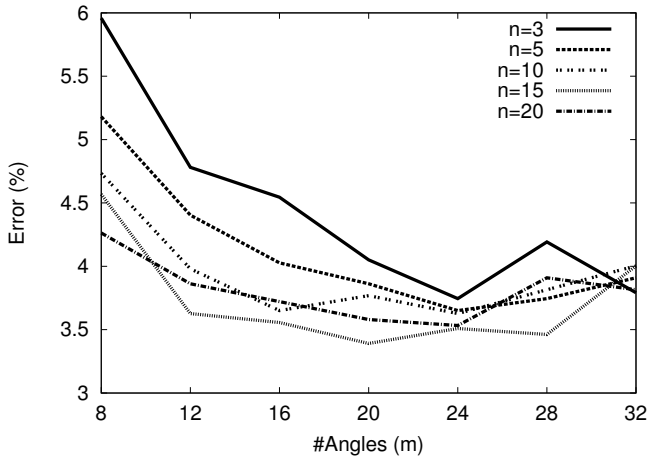


Figure 7.9: Fitting polar histogram parameters. Error for the best PCA dimension set for each m (*angles*) $\times n$ (*circles*) histogram is shown.

Given the good results for both feature types, which use quite different representations, a natural next step was to merge them. This combination led to small improvements in mean classification error to 2.7% (see Table 7.6). This is unlikely to be significantly different from the GEO_2 result, due to the larger standard deviation (0.29% vs. 0.16%).

7.4.3 Discussion

Table 7.7 shows the accumulated confusion matrix for one of the geometric features experimentation. As expected, most errors are produced in the classification of Right, Subscript and Superscript relationships, whereas Below and Inside relationships have few errors.

The SVM classifier is influenced by the prior probabilities of the classes in the training data (Right: 68.9%, Subscript: 5.9%, Superscript: 9.0%, Below: 12.1%, Inside: 4.1%). The Right relationship represents about 69% of the samples, and its recognition error was very low. The Superscript relation had a 6.3% error, but it is the Subscript relation that is most challenging, with more than 20% error: the Right/Subscript confusion is by far the most frequent.

Table 7.8 shows the confusion matrix for the shape-based features. The classification errors follow a very similar distribution. Errors in Subscript and

Superscript relations are slightly higher, as well as for Inside. The error rate for Below relationships in ground-truth is lower, but the classifier has more false positives for the Below relationship.

The polar histogram descriptor obtained results comparable to the geometric features when no symbol information is used, but was outperformed by the geometric features when typographic classes are used to move vertical centroids. One possible direction for future work is to try and incorporate

Table 7.7: Confusion matrix for geometric features (GEO_2) accumulated for 10 classification experiments. Ground-truth labels are shown along the rows (FN: false negative rate, FP: false positive rate).

	GEO_2 Output					
GT	Right	Sub	Sup	Below	Inside	**FN**
Right	28888	196	149	7	8	1.2%
Sub	498	1993		25	2	20.8%
Sup	239		3597	2		6.3%
Below	18	42	4	5083	9	1.4%
Inside	9				1707	0.5%
FP	2.6%	10.6%	4.1%	0.7%	1.1%	

Table 7.8: Confusion matrix for shape descriptors (SHP) accumulated for 10 classification experiments. Ground-truth labels are shown along the rows (FN: false negative rate, FP: false positive rate).

	SHP Output					
GT	Right	Sub	Sup	Below	Inside	**FN**
Right	28863	251	130	4		1.3%
Sub	581	1912		25		24.1%
Sup	322		3514	2		8.4%
Below	17	23	2	5114		0.8%
Inside	37	4		22	1653	3.7%
FP	3.2%	12.7%	3.8%	1.0%	0%	

this information into the shape descriptor, in order to take advantage of the categories information.

From the results, the proposed descriptors are not sufficient on their own for spatial relationship classification. Language models may be needed to distinguish cases where the geometric conditions represent different relations depending on the symbols involved (e.g. the right relation 'Px' vs. the subscript relation 'p_x').

However, there are opportunities to improve our descriptors, for example using continuous values for the bins in our polar histograms. For both feature types presented, it would be good to find better ways to identify the writing line, middle line (e.g. top of a lower-case 'x'), or a point between these in order to better handle the most common confusions (Right vs. Subscript or Superscript).

7.4.4 Summary

In this section we dealt with the classification of spatial relations between handwritten mathematical symbols and subexpressions. We presented a set of geometric features and a novel set of shape-based features. We tested different normalization factors for the geometric features, as well as modified the features using information about typographic categories. The histogram-based shape feature provides comparable results to geometric features when no information about symbol typographic categories (e.g. ascender) is used. The combination of both sets of features led to a small improvement in accuracy. Due to the higher computational cost of shape-based features and the obtained results, within this configuration, geometric features represent a best option for spatial relationship classification.

In future work, we will consider including symbol typographic classes into the shape-based feature representation. Also, it might be interesting to consider adding a rejection class, to detect when two subexpressions are unrelated. Finally, these features can be applied to printed expressions and compared with earlier work [Aly et al., 2009].

7.5 Recognition of Handwritten Math Expressions

In this thesis we have proposed an integrated model for recognizing mathematical expressions. We have evaluated features and classifiers for math symbols and spatial relationships. Below we assess the performance at expression level of the proposed approach for online handwritten mathematical expressions.

7.5.1 Experimental Setup

As discussed in Chapter 6, several issues have to be taken into account for reporting experimental results in mathematical expression recognition. For this reason, we used the public database of the last CROHME competitions (Section 7.1.2) and the results are reported using the set of metrics based on label graphs (Section 6.2.4). Furthermore, we developed `seshat`,[f] an open-source system that implements the parsing algorithm of the approach proposed in this thesis for online handwritten mathematical expressions (Chapter 5).

The symbol classifier implemented in `seshat` is the linear combination of RNNs with online and offline features, which provided the best results in handwritten symbol classification (Section 7.2). Spatial relationships classification was performed using geometric features and symbol categories (Section 4.2). The combination with shape-based features is not used because it would require more computation and the improvements were not significant. The remaining configuration of the parsing algorithm is described throughout this document: the symbol duration model (Section 3.1), the symbol segmentation model (Section 3.2) and the probabilistic grammar (Section 4.4).

7.5.2 CROHME 2013 Experiments

In order to provide comparable results we report experimentation using the datasets of the CROHME 2013 competition. As described in Section 5.4, we need a validation set for training the system. For this reason, we extracted 500 math expressions from the training set of the CROHME competition, such that both sets had the same distribution of symbols. Therefore, the sets used in the experimentation described in this section were a training set made up of $8,336$ expressions and a validation set with 500 expressions.

After training all the models and parameters (see Section 5.4), we classified the 671 online math expressions of the test set of the CROHME 2013 competition. Eight systems participated, including a preliminary version of this model (system IV). All but two of the systems used only the competition

[f] https://github.com/falvaro/seshat (more details in Section 8.1.2).

training set (8,836 math expressions). System III also used 468 additional math expressions, and System VII was trained using roughly 30,000 math expressions from a private corpus.

Table 7.9 shows the performance metrics at symbol level, and Table 7.10 shows results at stroke level. Results show that system VII performed the best, obtaining very good results. It was awarded the strongest system in the competition. However, as they used a large private dataset we were not able to fairly compare its performance to that of the other systems. System IV was a preliminary version of `seshat` and was awarded the best system trained using only the CROHME training dataset. The main differences between System IV and `seshat` are as follows: System IV only had the online RNN classifier and `seshat` combines it with an offline classifier; `seshat` uses *visibility* between strokes and a clustering-based penalty; and the probabilities of the grammar were not estimated in System IV.

The approach proposed in this thesis outperformed at all levels the other systems that were trained using the CROHME dataset. At symbol level, symbol classification of correctly segmented symbols of `seshat` obtained recall and precision of 82.2% and 81.0%, while the next best system (system VIII) obtained 73.8 and 71.0%. The absolute difference was more than 8% in recall and 10% in precision. Regarding spatial relationships, recall and precision for `seshat` stood at 88% and 82%, while System VIII gave 73% and 77.7%. This translates into an absolute difference of 15% in recall and 4.3% in precision. Results at stroke-level were also better than those of the systems trained only using the CROHME dataset. The systems were ranked according to the global error metric ΔE, where `seshat` had 13.2%, some 6.1% less than next best system (19.3%).

In addition to the experimentation comparing our proposed model to other systems, it is interesting to see how each of the stochastic sources contribute to overall system performance. For this reason we also carried out an experiment to observe this behaviour. Some models are mandatory for recognizing math expressions: the symbol classifier, the spatial relationships classifier and the grammar. We performed an experiment using only these models (base system), then adding the remaining models one by one. Also, the grammar initially had equiprobable productions and then we compared the performance when the probabilities of the rules were estimated. The order of introduction of the different sources was determined such that we first added the sources for symbol recognition and later the sources for the structural analysis of the mathematical expression. Also, seeking to avoid that the contribution to system performance of each source was hidden by other source.

Table 7.9: Object-level evaluation for the CROHME 2013 test set. Systems sorted by decreasing recall for correct symbol segmentation and classification (Seg+Class). It should be noticed that System IV is a preliminary version of `seshat`.

	Segments		Seg+Class		DAG Relations	
	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.
VII	97.9	98.1	93.0	93.3	95.2	95.5
seshat	**92.0**	**90.7**	**82.2**	**81.0**	**88.0**	**82.0**
IV*	85.0	87.1	73.9	75.8	76.3	79.9
VIII	90.3	86.9	73.8	71.0	73.0	77.7
V	84.5	86.5	66.7	68.3	72.6	74.3
II	80.7	86.4	66.4	71.1	45.8	63.0
III	85.2	77.9	62.6	57.3	88.5	78.3
VI	57.9	47.3	47.7	39.0	31.8	70.0
I	46.9	38.4	25.2	20.6	33.7	71.6

Table 7.10: Stroke-level evaluation for the CROHME 2013 test set. Systems sorted by increasing ΔE. ΔB_n and ΔE are measured on directed labels graphs (8548 strokes; 81007 (undirected) stroke pairs). It should be noticed that System IV is a preliminary version of `seshat`.

	Label Hamming Distances				μ error (%)	
	Strokes	Pairs	Seg	Rel	ΔB_n	ΔE
VII	537	1777	170	1607	2.4	4.3
seshat	**1583**	**7829**	**700**	**7129**	**6.9**	**13.2**
IV*	2187	9493	1201	8292	10.1	18.3
VIII	2302	15644	4945	10699	12.1	19.3
II	2748	19768	1527	18241	13.9	22.0
V	2898	10803	1228	9575	12.7	22.8
III	3415	15135	1262	13873	15.0	26.2
VI	4768	43893	5094	38799	27.6	36.7
I	6543	41295	5849	35446	26.8	41.6

Tables 7.11 and 7.12 show the changes on system performance when each source of information is added. Global error metrics ΔB_n and ΔE consistently decreased with each added feature. Symbol segmentation and symbol recognition also improved with each addition. It is interesting that, when no segmentation model was used, symbol segmentation still gave good results. This was the case because the parameters of the integrated approach converged to high values of the insertion penalty and low values of the segmentation distance threshold. In this way, the parameters of the system itself could alleviate the lack of a segmentation model. In any case, when the segmentation model was included, system performance greatly improved. Furthermore, we would like to remark that, when the relations penalty was included in the model, the number of hypotheses explored was reduced by 56.7%.

The structural analysis is harder to evaluate than symbol recognition. Prior to estimating the grammar probabilities, the results at object level seem to worsen when the segmentation model was included, although at stroke-level the errors in spatial relationships decreased from about $11,000$ to $9,500$ stroke pairs. Because of the inheritance of spatial relations in label graphs Zanibbi et al. [2011], some types of structural errors can produce more stroke-pair errors than others Zanibbi et al. [2013]. Specifically, when the segmentation model was used, the segmentation distance threshold was approximately twice as high as the value of the threshold in the base system. This had two effects. First, that the system was able to account for more symbol segmentations, as shown by the corresponding metrics. Second, that a bad decision in symbol segmentation can lead to worse structural errors. Nevertheless, the estimation of the probabilities of the grammar led to great improvements in the detection of the structure of the expression with barely any changes in symbol recognition performance.

Table 7.11: Contribution to overall system performance at object level of the different sources of information used. The models and features listed in each row are cumulative, such that the system shown in the last row includes all information sources.

	Segments		Seg+Class		Relations	
	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.
Base system	87.6	83.6	78.3	74.7	87.0	75.0
+ duration model	88.0	84.2	78.7	75.3	86.2	75.5
+ segmentation model	90.9	90.4	81.3	80.8	81.5	73.7
+ relations penalty	91.8	91.3	82.0	81.5	81.4	74.0
+ grammar estimation	92.0	90.7	82.2	81.0	88.0	82.0

Table 7.12: Contribution to overall system performance at stroke level of the different sources of information used. The models and features listed in each row are cumulative, such that the system shown in the last row includes all information sources.

	Label Hamming Distances				μ error (%)	
	Strokes	Pairs	Seg	Rel	ΔB_n	ΔE
Base system	1987	11999	1172	10827	9.1	16.5
+ duration model	1928	12055	1138	10917	9.0	15.8
+ segmentation model	1634	10272	834	9438	7.6	13.7
+ relations penalty	1556	10397	724	9673	7.5	13.2
+ grammar estimation	1583	7829	700	7129	6.9	13.2

7.5.3 CROHME 2014 Experiments

The last two editions of the CROHME competition share the same training set (Section 7.1.2). In previous section we performed a detailed experimentation using the test set of the CROHME 2013 competition in order to evaluate the approach developed in this thesis and the diferent statistical models proposed. Afterwards, once our approach has been validated, we participated in CROHME 2014 competition [Mouchère et al., 2014]. Below we detail our participation and analyze the results.

We trained `seshat` using the train set of the CROHME competition (8,836 math expressions) and for validation we used the test set of the previous CROHME 2013 competition (671 math expressions). This way we could use the entire train set for estimating the different models required for the recognition system.

Seven systems participated in mathematical expression recognition task of the CROHME 2014 competition. Six of them had already participated in previous edition. All the systems were trained using only the train data of the competition, except for System III that was trained on a private corpus collected from writers in several countries.

The results of the math expression task of the competition at object level are shown in Table 7.13 and results at stroke level are shown in Table 7.14. System III was System VII in CROHME 2013 competition, and again they obtained the best results and it was awarded the best system of the CROHME 2014 competition. Unfortunately, as they use private data for training the comparison cannot be done under the same conditions. Our proposal, `seshat`, obtained the best results among the remaining systems and it was awarded the best system using only the CROHME dataset. At object level, `seshat` symbol recognition results (correctly segmented and classified) obtained 8% higher recall and 7% higher precision than next best performance (recall of System V and precision of System IV). The structural evaluation at object-level shows that our approach had more than 12% higher recall and 7% higher precision than next best results (recall of System VII and precision of System VI). Finally, at stroke level, `seshat` also outperformed the systems trained using only the training set of the competition. The values of the global performance error metrics ΔB_n and ΔE (see Section 6.2.4) for `seshat` were 6.2% and 12%, respectively. System VII obtained the next best results with 9.4% and 17.7%, such that our system had 3.2% lower ΔB_n error and 5.7% lower ΔE error.

In addition to the mathematical expression recognition task, two additional tasks were proposed in CROHME 2014 competition. First, an isolated symbol recognition task whose results have been reported in Section 7.2.3. Second, a

Table 7.13: Object-level evaluation for the CROHME 2014 test set. Systems sorted by decreasing recall for correct symbol segmentation and classification (Seg+Class).

	Segments		Seg+Class		DAG Relations	
	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.
III	98.4	98.1	93.9	93.6	94.3	94.0
I-seshat	**93.3**	**90.7**	**86.6**	**84.2**	**84.2**	**82.0**
V	88.2	84.2	78.5	74.9	61.4	72.7
IV	85.5	86.1	76.6	77.2	70.8	71.5
VII	89.4	86.1	76.5	73.7	71.8	71.7
VI	83.1	85.4	69.7	71.7	66.8	74.8
II	76.6	80.3	67.0	70.2	60.3	63.7

Table 7.14: Stroke-level evaluation for the CROHME 2014 test set. Systems sorted by increasing ΔE. ΔB_n and ΔE are measured on directed labels graphs (13796 strokes; 288660 (undirected) stroke pairs).

	Label Hamming Distances				μ error (%)	
	Strokes	Pairs	Seg	Rel	ΔB_n	ΔE
III	845	2489	836	1653	2.2	4.6
I-seshat	**2026**	**7174**	**2540**	**4634**	**6.2**	**12.0**
VII	3558	11795	4388	7407	9.4	17.7
V	2742	13477	4386	9091	10.3	18.2
IV	3406	12768	5328	5328	10.0	19.2
VI	4039	13325	5670	7655	10.1	19.4
II	4523	17449	7772	9677	14.4	25.8

new task dealing with the recognition of matrices, a domain that can be considered as math expressions with nested subexpressions arranged in a tabular layout (rows and columns). The organizers provided a training set composed of 256 online mathematical expressions containing 362 matrices. They also provided an extended version of the label graph evaluation metrics that take into account matrix structures. The test set had 122 expressions containing 175 handwritten matrices.

Although there were eight participants in the competition, only two of them submited a system for matrix recognition: System III and our system (I). System III used the same system than the trained for the expression recognition task with a specific post-processing to retrieve the rows and columns. We also used the system trained for the expression recognition taks with the following considerations in order to account for matrices. First, the number of rows and columnns must match when combining hypotheses in order to create a matrix. Second, the spatial relationships between rows or columns take into account the differences between the centers of each row/column. Specifically, when two columns are combined, feature 'D' is averaged for each pair of row centers; and when two rows are combined, feature 'dhc' is averaged for each pair of column centers (features are shown in Fig. 4.1). Finally, the grammar was extended with productions in order to account for matrices.

Table 7.15 shows the results of the matrix recognition task of the CROHME 2014 competition. System III obtained significantly better results in all metrics, where only symbol recognition rate obtained similar results. Comparison has to take into account that System III was not trained with the data of the competition. Results of both systems show that mathematical expression recognition is harded when matrices are considered, given that the performance metrics are lower in this task. As the CROHME organizers pointed out, there is a great difference in the detection of columns with respect to row detection. This is because sometimes is difficult to decide when a horizontal concatenation of symbols represent a subexpression or it has to be split into different columns.

Although this was our first approach to this task and it would require further research, this experimentation validated the model developed in this thesis also for matrix recognition. Furthermore, despite the general experimentation has been carried out in the domain of the expressions considered in the CROHME datasets, this extension for recognizing complex structures like matrices proves that the formal framework developed in this thesis can be extended to account for other structures of math notation.

Table 7.15: Results of the matrix recognition task of the CROHME 2014 competition.

	System I (`seshat`)	System III
Expression Rate	31.2	53.3
Symbol Recognition Rate	87.4	89.8
Matrix Recall Rate	73.1	92.6
Row Recall Rate	70.6	92.0
Column Recall Rate	50.8	69.2
Cell Recall Rate	55.4	71.1

7.5.4 Summary

The experimentation reported in this section validates the approach presented in this thesis for online handwritten mathematical expression recognition. We developed the software that implements the proposed system and released it as open-source. Furthermore, we report results using standar performance metrics and the large public dataset used in the CROHME international competition. In this way, we provide comparable results in order to help the progress in this research domain.

A detailed discussion of system performance and the contribution of each part of the model is also provided. The system implemeting our approach outperforms other approaches at all levels. It was awarded best system using only the training dataset of the last two CROHME competitions [Mouchère et al., 2013, 2014].

7.6 Recognition of Printed Math Expressions

The approach and the statistical framework presented in this thesis is able to account for any type of mathematical expression. Although most of the effort has been done on online handwritten mathematical expressions, we wanted to validate the application of this model to offline printed math expression recognition. Consequently, below we report an experiment to validate the proposed approach for recognizing printed math expressions.

7.6.1 Experimental Setup

As discussed along this document, the main difference for recognizing online or offline mathematical expressions are the primitives of each type of expression. The input sequence of observations $\boldsymbol{o}$ is a sequence of strokes in online recognition, and an enumeration of connected components (in any order) in offline recognition. However, there is a main obstacle for applying this approach to offline recognition: the ground-truth information.

In this experiment we used the INFTY database (Section 7.1.1), a public large resource for printed math expression recognition. This dataset has many ground-truth information at several levels, but the symbols are annotated with their bounding box coordinates. Although these coordinates can precisely locate a symbol within an image, in several cases the region of the symbol can contain parts of other symbols (see Fig. 7.10). Therefore, as our models require connected components as primitives, the ground-truth of the INFTY dataset has to be processed in order to obtain the information at that level.

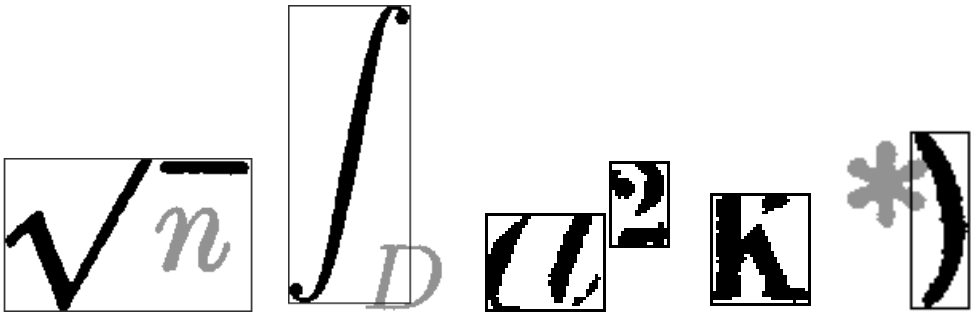


Figure 7.10: Examples of problems with the INFTY ground-truth information in order to isolate the connected components from the bounding box coordinates.

Extracting the connected components for each symbol given its bounding box is not straightforward. One option would be to take into account the bounding box of all the symbols within an expression, and obtain the best mapping from symbols to components. Another local method would be using some heuristics, e.g. detecting the larger component in the region. But these methods are error-prone because there are touching symbols, some samples might be split in several components (and small components should not be removed), square roots can contain large symbols, etc. In the end, manual supervision is required in order to guarantee that ground-truth at connected component level is correctly obtained in all cases. When we performed the printed recognition experiment reported in Section 7.3, we discarded touching symbols, and we used local heuristics with manual supervision in order to label the samples of symbols for the experimentation.

Another problem derived from the annotation using bounding boxes, is that we cannot use the label graph performance evaluation metrics (Section 6.2.4). The tools for computing these metrics work at stroke level, although Zanibbi et al. [2011] defined the metrics at primitive level such that other primitives could be used (e.g. pixels or connected components). Again, we would need the ground-truth at connected component level in order to generate label graphs from annotated data, and tools that work with this type of offline data.

Even though the INFTY database (Section 7.1.1) was released several years ago, comparison of different approaches for printed recognition is difficult due to the problems of performance evaluation in this field (Chapter 6) and because there are no standard train and test set partitions. We randomly selected 3K math expressions from the INFTY database as a test set. The remaining expressions were used for training the system, except for 500 expressions that were extracted at random for validation. We modified `seshat` (Section 8.1.2) to make it able to process offline expressions. Only two changes are necessary: the system has to read images and compute connected components; and the symbol classifier and feature extraction has to be specific for printed symbols.

Below we present a pilot experiment of printed mathematical expression recognition. The goal of this study is to validate our approach in this scenario, but further experimentation should be carried out with the required ground-truth information and evaluation tools. The configuration of the `seshat` system adapted for printed recognition was the following. The symbol recognizer was a simple nearest neighbor classifier as described in the experiment of Section 7.3. This classifier does not need to be trained and presented a good performance for this task. The symbol duration model was obtained from the

number of connected components of the train symbol samples. The segmentation model and the spatial relationship classifier estimated from the CROHME train set were directly used in this pilot experiment. Finally, we manually defined a 2D-PCFG that accounted for the 183 mathematical symbols considered, and the structures of the expressions that appear in the INFTY database. The productions of the grammar were equiprobable.

7.6.2 Results and Discussion

In this section we present the results of the pilot experiment of printed mathematical expression recognition. The performance evaluation at symbol level was computed such that, if the bounding box coordinates of the recognized symbol match the ground-truth coordinates, it is properly segmented. If the symbol class of the properly segmented symbol is also correct, then the symbol is correctly segmented and classified. We tuned the parameters of the parsing algorithm maximizing the rate of symbols correctly segmented and classified when recognizing the validation set. Finally, we classified the test set with this system and the obtained results are reported in Table 7.16.

Table 7.16: Recognition results at different levels of 3K printed mathematical expressions from the INFTY dataset.

Symbol Segmentation Rate	97.8%
Symbol Seg. + Classification	92.9%
Expression Recognition Rate	45.0%
IMEGE mean	12.4%
EMERS mean	2.3

Symbol segmentation and symbol recognition results obtained a very good performance, because printed expressions are more regular than handwritten expressions, and printed symbol classifiers provide a lower recognition error than handwritten symbol classifiers.

On the other hand, as we could not compute label graphs error metrics, we reported two proposals for global evaluation. IMEGE is an image-based global error metric described in Section 6.2.3, and EMERS is an edit distance between trees described in Section 6.2.2. We report the mean values of the global metrics, and also the histograms of the recognition errors of the samples in the test set (Fig. 7.11) because the distribution of the errors is not normal.

Analyzing the histogram of the errors we can see that both metrics present a very similar distribution, although the range is different because IMEGE is a normalized error while EMERS is an edit distance. The EMERS values represent the edit distance between the recognition result and the ground-truth, thus we can see how most of the samples have few errors and the number of expressions requiring many edit operations is very low. Regarding IMEGE, as it is a normalized error metric, the distribution is slightly different. We can see how either expressions are perfectly recognized (zero error) or most of the global errors are in the range $10\% - 30\%$. It seems reasonable, since a small global error would require few errors in large expressions.

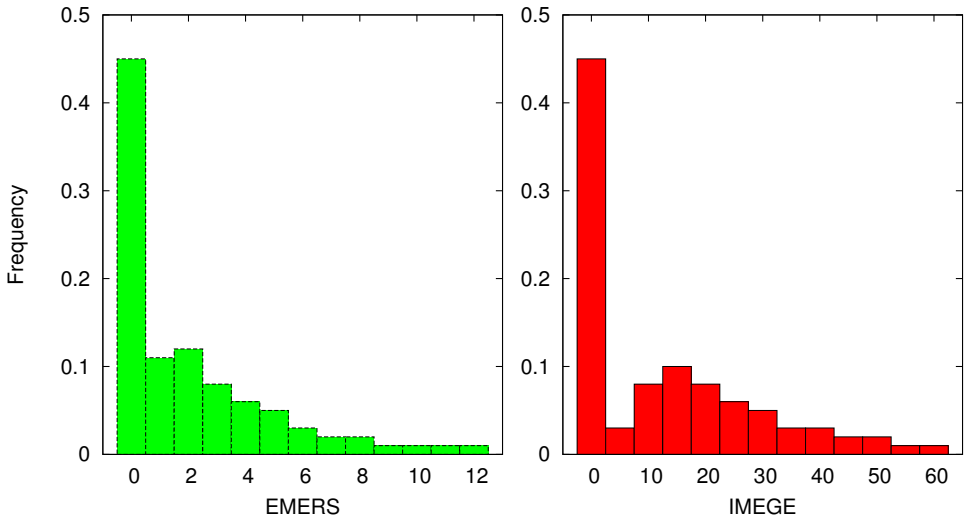


Figure 7.11: Histogram of error metrics of the expressions in the test set for the printed mathematical expression recognition experiment.

In Chapter 6 we presented a discussion about problems in performance evaluation. One of the main problems was measuring structural and global errors due to representation ambiguity. In this experimentation we reported two global metrics, and both metrics reported *practically* the same number of expressions with no error. The IMEGE metric dealt with ambiguity because it is based on the image representation computed from the LaTeX strings. On the other hand, EMERS computes a tree edit distance, but we obtained the MathML trees from the LaTeX strings and most of the ambiguities were removed [Sain et al., 2010].

We report the expression recognition rate as the percentage of expressions with IMEGE error equals to zero (equivalent to EMERS equals to zero). There were only 5 expressions out of 3000 of the test set that had IMEGE equals to zero but EMERS greater than zero. In all these cases, the LaTeX expressions contained subscript or superscript relationships with different bracketed *base* of the relation. The IMEGE graphic representation was equivalent but the generated trees were different, producing that EMERS reports the expressions as incorrect (see Fig. 7.12).

	Recognition	Ground-truth
LaTeX	`P {( x )}^{s}`	`P(x)^{s}`
Render	$P(x)^s$	$P(x)^s$

MathML	`<mathml>`
	`<mrow>`
	`<mi>P</mi>`
	`<msup>`
	`<mrow>`
	`<mo>(</mo>`
	`<mi>x</mi>`
	`<mo>)</mo>`
	`</mrow>`
	`<mi>s</mi>`
	`</msup>`
	`</mrow>`
	`</mathml>`
		`<mathml>`
		`<mrow>`
		`<mi>P</mi>`
		`<mo>(</mo>`
		`<mi>x</mi>`
		`<msup>`
		`<mo>)</mo>`
		`<mi>s</mi>`
		`</msup>`
		`</mrow>`
		`</mathml>`

Error	IMEGE = 0;	EMERS = 2.5

Figure 7.12: Example of evaluation difference between EMERS and IMEGE due to representation ambiguity.

7.6.3 Summary

Mathematical notation is well-defined, independently of the type of expression considered for automatic recognition: printed or handwritten, online or offline. The approach and statistical framework developed in this thesis, with specific treatment for symbol recognition, have proven to recognize printed mathematical expressions. In this section, we reported a pilot experiment that validates

this approach. Further experimentation need to be done with ground-truth information at level of connected components. Also, all the probabilistic models integrated in the system and the grammar estimation should be calculated from the set of printed expressions. Finally, under these proper conditions, standard partitions of the INFTY dataset would be very useful for comparison of different proposals in this field.

Chapter

8 Applications

The motivation of this thesis was to develop an approach for mathematical expression recognition. This problem has several applications, and the techniques explored during the research process can also be useful in order to tackle other related situations. Throughout the development of this study, we have created several tools and analyzed different applications.

In this chapter we present the main applications of mathematical expression recognition we have explored. First, in Section 8.1 we review the open-source software that has been published during the development of this thesis. Second, in Section 8.2 we show an application for introducing online handwritten math expressions into computers and information retrieval. Then, in Section 8.3 we present μCAPTCHA, a novel application of human interaction proof based on math recognition. Finally, in Section 8.4 we describe a method for layout analysis of structured documents based on parsing 2D-PCFG.

Chapter Outline

8.1 Open-source Software

As a result of the developments of this thesis, different open-source software has been made publicly available. We hope that this will help the progress in mathematical expression recognition. Below we briefly review the software released.

8.1.1 Printed Math Expression Recognition

As described in Section 2.1, initially we dealt with the problem of recognizing printed math expressions. The approach described in this thesis was obtained after some years of developments starting from that model. This initial parser for printed expressions was published in an international conference [Álvaro et al., 2011] and obtained a good performance. We released the source code of the recognizer and referenced it within the paper. The software can be obtained at

`https://github.com/falvaro/pme_parser`

and it is licensed under the *GNU General Public License version 3.0 (GPLv3)*.

The software accepts an image as input thanks to the Magick++ library,[a] and outputs the recognized expression in LaTeX format. The symbol classifier is a simple nearest neighbor classifier as described in Section 3.4. A sample grammar is provided and we manually defined probability functions based on geometric features for spatial relationship classification.

8.1.2 Handwritten Math Expression Recognition

In this thesis we developed a formal approach for recognizing mathematical expressions. We released the software that implements the system for recognizing online handwritten math expressions according to the proposed methodology. The source code is publicly available at

`https://github.com/falvaro/seshat`

and it is licensed under the *GNU General Public License version 3.0 (GPLv3)*.

The software accepts input in InkML format or SCGINK format (plain text), and outputs the recognized expression in LaTeX and InkML (which includes MathML). As detailed along this thesis, there are several distributions inside a math recognition system, such that `seshat` contains the configuration

[a] `http://www.imagemagick.org/Magick++/`

described in experiments of Section 7.5.3. It is worth to mention that symbol classifiers are BLSTM-RNNs, such that integrates part of the code of the RNNLIB library,[b] and the Xerces-c library[c] is used for parsing InkML.

This software was awarded "Best system trained on CROHME dataset" in the CROHME 2014 competition [Mouchère et al., 2014].

8.1.3 Features for Handwriting Recognition

Classification of handwritten math symbols is a challenging task. For this reason, we explored several features as described in Section 3.3. The code for extracting the online features, hybrid features and four different sets of offline features (PRHLT, FKI, RWTH and Polar), is available at the software section of the PRHLT research center website

`www.prhlt.upv.es`

All the tools for computing the features are released under the *GNU General Public License version 3.0 (GPLv3)*.

8.1.4 Image-based Evaluation

Regarding the problem of automatic performance evaluation in mathematical expression recognition, we developed a metric based on image matching. The code for computing the IMEGE error metric (see Section 6.2.3) is available at

`https://github.com/falvaro/imege`

and it is licensed under the *GNU General Public License version 3.0 (GPLv3)*.

The software includes the code for computing the BIDM algorithm and a wrapper for computing the IMEGE error. It computes the error value directly from two given LaTeX expressions. The `l2p`[d] tool is used in order to obtain the image representation from the LaTeX string.

[b] `http://sourceforge.net/projects/rnnl/`
[c] `http://xerces.apache.org/xerces-c/`
[d] `http://redsymbol.net/software/l2p/`

8.2 LaTeX Transcription and Information Retrieval

One of the first applications of mathematical expression recognition that a person would think about is introducing math notation into a computer using handwriting. It is very useful in order to obtain the transcription into a certain notation, even more if some encodings are unknown. Therefore, we made available an online demonstrator of online handwritten mathematical expression recognition at

http://cat.prhlt.upv.es/mer/

The web demo accepts handwriting as input and provides the LaTeX transcription of the recognized math expression, such that the underlying system is based on `seshat`. Furthermore, the expression can be used to retrieve relevant documents from internet using Google's search engine, or mathematical information can be obtained thanks to Wolfram Alpha online service.[e] An example of the interface is shown in Fig. 8.1, such that the recognized expression is used in order to retrieve relevant documents using Google's search engine.

Mathematical information retrieval is an active research field and specific techniques have been proposed [Zanibbi and Blostein, 2012]. Currently, we simply search for results using the LaTeX string as query and Google's engine, but specific techniques and metrics are being developed [Stalnaker and Zanibbi, 2015] and it would be interesting to used them in the future.[f]

[e] https://www.wolframalpha.com/

[f] An example of information retrieval using more advanced techniques developed by the DPRL can be found at http://saskatoon.cs.rit.edu/tangent

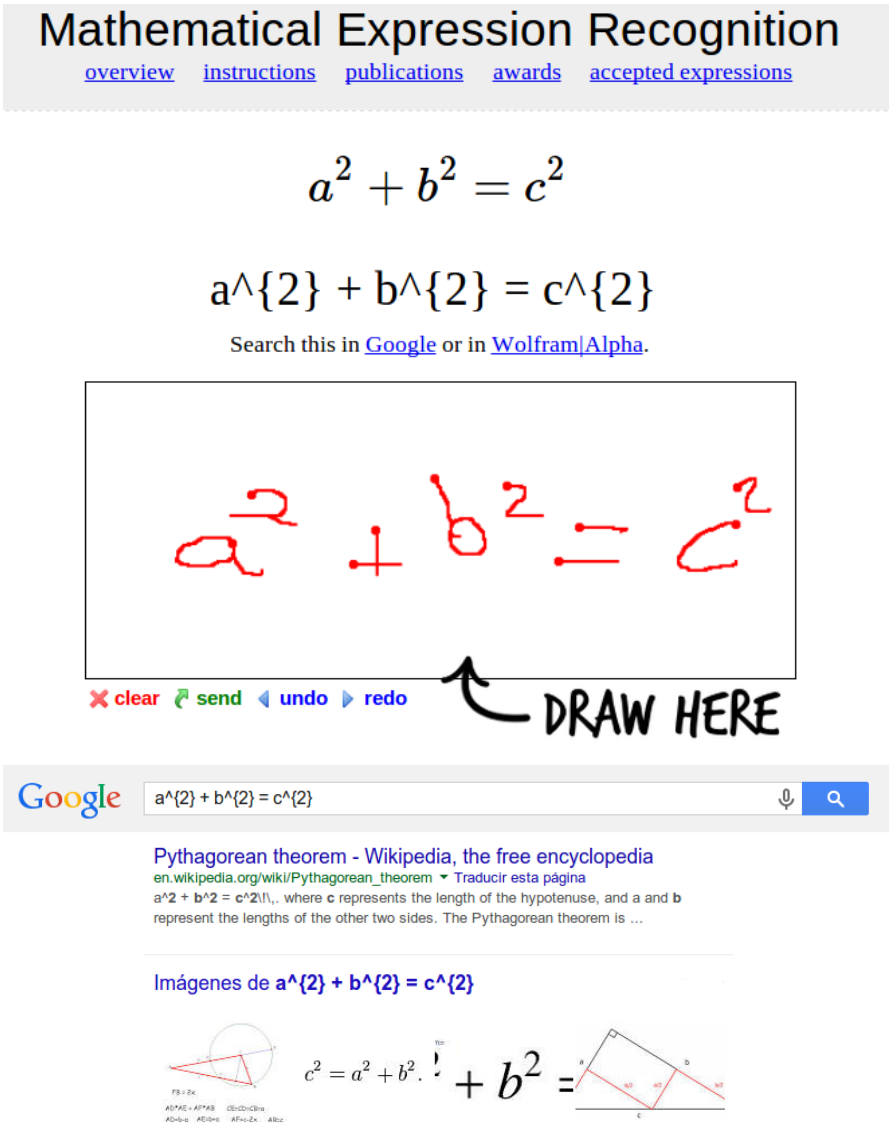


Figure 8.1: Example of the online demo for LaTeX transcription and information retrieval of online handwritten mathematical expressions.

8.3 μCAPTCHA: Math-based CAPTCHA

A captcha (Completely Automated Public Turing Test to Tell Computers and Humans Apart) is a program that protects online services against bots by generating and grading challenges that humans can pass but computers cannot. Online services are often protected with captchas that typically must be solved by typing on a keyboard. Now that smartphones and tablets are increasingly being used to browse the web, new captchas best suited to touch-capable devices should be devised, since entering text on soft keyboards is usually uncomfortable.

We contribute to solving this issue with μCAPTCHA, a novel method to tell humans and computers apart by means of math handwriting input. Below, we first introduce the problem in Section 8.3.1 and we review the different captchas that have been proposed in the literature in Section 8.3.2. Then, in Section 8.3.3 we describe μCAPTCHA, our captcha proposal based on handwritten math expression recognition. In Section 8.3.4 we report the experimentation we carried out to validate the method. Finally, we summarize the properties of μCAPTCHA and present conclusions in Section 8.3.5.

8.3.1 Introduction

Captcha belongs to the set of protocols called HIPs (Human Interactive Proofs), which allow a person to authenticate as belonging to a select group [Rusu and Govindaraju, 2004]; e.g. human as opposed to machine, adult as opposed to a child, etc. Captchas are used on the web for many purposes, such as to prevent massive email account creation, avoid spam comments in blogs and forums, or verify financial transactions. The main advantage of captchas is that they operate without the burden of passwords, biometrics, mechanical aids, or special training [Baird and Popat, 2002]. However, the usability of captchas is a subject of intense debate [Bursztein et al., 2014; Chellapilla et al., 2005; Yan and El Ahmad, 2008].

Historically, users have had to deal with captchas in the form of images of distorted text, such distortions based on the weaknesses of Optical Character Recognition (OCR). However, as OCR software improves, solving captchas is becoming increasingly difficult. This places a burden on users [Bursztein et al., 2014], who are progressively reluctant to solve them [Shirali-Shahreza and Shirali-Shahreza, 2006]. Moreover, the majority of captchas are designed for use on computers and laptops, which do not align well with the interaction style of mobile users (see Fig. 8.2).

According to the international telecommunication union, there are more

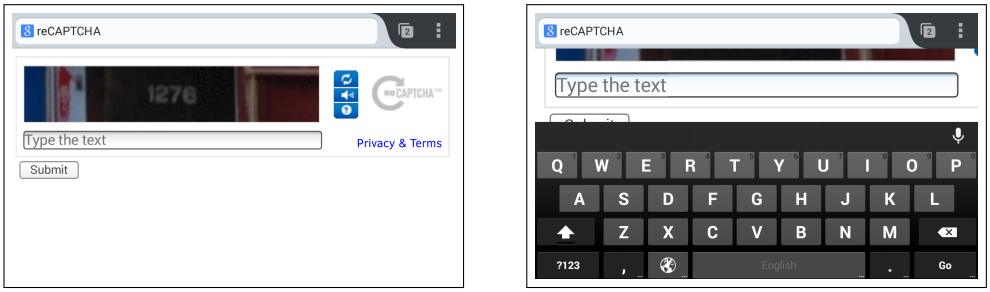


Figure 8.2: Solving captchas on a mobile device is rather uncomfortable. For instance, focusing on a text field causes zooming and field positioning which do not allow for the captcha to be read properly.

than 2 billion active mobile-broadband subscriptions worldwide.[g] An independent study by comScore[h] confirmed that in 2013 smartphones and tablets surpassed desktop PCs to become the leading platform in terms of total time spent online, either via web browsers or apps that make use of web services. These figures urge for a prompt revision on the design of HIPs in general and captchas in particular, since entering text in soft keyboards is uncomfortable and error-prone [Chen et al., 2010]. To this end, drawing is presumably easier and quicker than typing on a mobile device [Kienzle and Hinckley, 2013].

We have developed μCAPTCHA, a novel way to tell humans and computers apart by handwriting mathematical expressions on a touch-capable device. The main advantage of μCAPTCHA is that it is language-independent, so it is equally easy to learn for everyone. Another potential advantage is that μCAPTCHA uses a controlled vocabulary of math symbols, which provides high recognition accuracy. Most important, the spatial relations between symbols (e.g. superscripts, subscripts, fractions, etc.) allow for a large number of different expressions to be generated. Furthermore, very few OCR software can recognize mathematical expressions. Together with its segmentation-resilient foreground noise technique (Fig. 8.3), μCAPTCHA should keep regular attackers at bay. Finally, as a byproduct of solving a μCAPTCHA challenge, a labeled handwritten mathematical expression is obtained. Therefore, μCAPTCHA contributes to building valuable machine learning datasets. Ultimately, this work informs our understanding of designing better web security measures.

[g] http://www.itu.int/ITU-D/ict/statistics
[h] http://www.comscore.com/mobilefutureinfocus2013

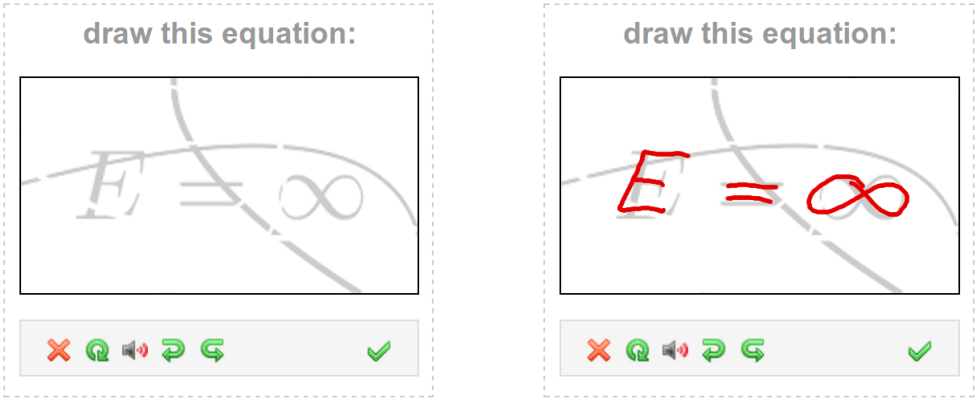


Figure 8.3: μCAPTCHA interface. A math expression is shown to the user (left), who has to draw it on a canvas (right). Buttons from left to right: clear strokes, request a new challenge, listen challenge (to write it in plain text), undo last stroke, redo last stroke, submit challenge.

8.3.2 Related Work

Many captchas are known to be broken, so their general strength is an area of increasing concern. There are very comprehensive surveys in this area [Baird and Popat, 2002; Basso and Bergadano, 2010; Hidalgo and Alvarez, 2011; Roshanbin and Miller, 2013]. The next generation of captchas are likely to be more difficult and awkward for human users; e.g. Rusu et al. [2010] combined handwritten text images with a random tree structure and random test questions that leveraged unique features of human cognition. This approach was found to be hard to solve by machines, but also by regular users.

Text-based CAPTCHAs

Text-based captchas obfuscate OCR by introducing image degradations; the harder ones provide better performance but are also harder for humans to solve [Chew and Baird, 2003; Coates et al., 2001]. Among others, reCAPTCHA [von Ahn et al., 2008] stands out as the most popular solution. It makes positive use of human effort by channeling the time spent solving captchas into digitizing text, annotating street imaginery, or building machine learning datasets. It also can be broken with 99% of accuracy [Goodfellow et al., 2014]. There is a large body of additional work on text-based captcha, and comprehensive surveys can be found in Baird and Popat [2002]; Basso and Bergadano [2010]; Hidalgo and Alvarez [2011]; Roshanbin and Miller [2013].

Image-based CAPTCHAs

Since most text-based captchas are vulnerable, researchers have proposed alternatives to character recognition. A popular one is the form of image recognition, which requires users to identify simple objects in the images presented. However, the need for a human to label the pictures in a large database is mandatory, so that answers can be verified. Examples in this regard include: Sketcha [Ross et al., 2010], Implicit CAPTCHA [Baird and Bentley, 2005], GOTCHA [Blocki et al., 2013], Asirra [Elson et al., 2009], or CAPTCHA-Zoo [Lin et al., 2011].

In visualcaptcha[i] the user has to click on the icon that best represents the word given. Currently, 37 icons and 20 audios are used as input stimuli. KittenAuth[j] also uses relatively small image databases. An image database small enough to be manually constructed is also small enough to be manually reconstructed by an attacker.

An interesting image-based captcha requires the user to rotate images to the correct orientation [Gossweiler et al., 2009]. An advantage of this idea is that it uses unlabeled images. One drawback is that images have to be carefully selected since certain images can have several correct orientations. Also, if the images display faces, they can be automatically detected and rotated.

CAPTCHA Alternatives

The Math CAPTCHA [Hernandez-Castro and Ribagorda, 2010] presents the user with an equation that must be solved. The number of different math challenges is very few and the answer for them is a single digit, so the challenge can be passed with trial and error. Further, this captcha is too complicated for the general public. There are other math-based captchas much simpler than this, relying on basic arithmetic operations shown in plain HTML, which can be automatically solved using regular expressions.

Another alternative is presenting the user with a short video, who can either describe it [Kluever and Zanibbi, 2009] or select the most appropriate answer from an option list [Shirali-Shahreza and Shirali-Shahreza, 2008]. These approaches do not scale well, as they require human intervention to enlarge the challenge database. In NuCaptcha[k] the user has to type moving letters, an approach that has been recently broken [Xu et al., 2012].

[i] http://visualcaptcha.net
[j] http://thepcspy.com/kittenauth/
[k] http://nucaptcha.com

Another option is to make the user play a game. PlayThru[l] provides different possibilities in this regard. Further, they claim to adopt a sophisticated mechanism to differentiate human game playing activity from automated activity [Mohamed et al., 2014]. However, there is evidence that these games can be easily spoofed.[m]

CAPTCHAs for Mobile Devices

A number of research efforts are aimed to simplify and speed-up captcha solving on mobile devices. TapCHA [Jiang and Tian, 2013] presents the user with a set of geometric shapes (square, triangle, etc.), and the challenge consists in dragging one of the shapes. Chow et al. [2008] propose to click on three valid English words out of a grid of 3x4 captchas. Some companies like Uniqpin[n] and ConfidentCAPTCHA[o] use this technique with images. Drawing CAPTCHA [Shirali-Shahreza and Shirali-Shahreza, 2006] presents the user with a large number of squares randomly drawn, and the user must connect three diamonds via taps. QapTcha[p] is a draggable component for web forms; users just have to move a slider to confirm that they are humans. All of these approaches can be broken with machine learning techniques and client-side scripting [Chellapilla and Simard, 2005; Lin et al., 2011].

SeeSay and HearSay [Shirali-Shahreza et al., 2013] allow the user to solve a captcha by submitting audio instead of text. There are, however, a number of situations where it is too noisy or inappropriate to use speech-based input. Moreover, recognition errors are one of the major concerns for users to accept these captchas [Shirali-Shahreza et al., 2013].

Handwriting CAPTCHAs

One option to ease text entry on mobile devices is by means of handwriting input. In Highlighting CAPTCHA [Shirali-Shahreza and Shirali-Shahreza, 2011] the user must trace an obfuscated word with a stylus. This method is similar in spirit to ours; however it has two drawbacks. First, it requires the user to *precisely* trace each character, which is difficult to perform on a mobile device due to the inaccuracy of user input [Hourcade and Berkel,

[l] http://areyouahuman.com
[m] http://spamtech.co.uk/software/bots/cracking-the-areyouhuman-captcha
[n] http://uniqpin.com
[o] http://confidentcaptcha.com
[p] http://myjqueryplugins.com/jquery-plugin/qaptcha

2008]. Second, the handwriting recognition that validates user input relies on heuristics that do not achieve competitive accuracy.

Regular text handwriting could be further explored as a means to solve captchas on mobile devices. However, recognition of cursive hand-made text is language-dependent and challenging both for computers and humans [Rusu et al., 2010]. Isolated handwritten character recognition could be used instead, but the number of symbols should be reduced in order to remove ambiguities (e.g. c, C, o, O, 0, etc.), resulting in few combinations available. Further, this is actually a subset of math symbols, which are language-independent and can be controlled together with the type of expressions shown to the user.

Another work closely related to ours is MotionCAPTCHA.[q] It presents the user with an image of a unistroke gesture that the user must draw. This is actually a proof of concept, and so it has a number of important security flaws. For example, the gesture vocabulary is very small (16 gestures, 4 bits of information entropy per challenge) and the solution to the captcha is available in the HTML source code.

8.3.3 System Design

The specific implementation of μCAPTCHA reflects a number of design principles. We believe that understanding these will be useful to others trying to build similar systems. Ideally, desirable properties of captchas are the following [Hernandez-Castro and Ribagorda, 2010]:

- Automation: the challenge should be automatically generated and graded by a computer program.
- Easy of use: the challenge should be taken quickly and easily by humans.
- High reliability: it should accept virtually all human users.
- Low false negatives: it should reject very few human users.
- Low false positives: it should reject virtually all machine users.

However, no matter the challenge, the whole system must be secure. Our design of μCAPTCHA validates what is submitted by the user on the server side, and is protected against brute-force attacks, replay attacks [Mohamed et al., 2014] and side-channel attacks [Zelkowitz, 2001]. Fig. 8.4 provides an overview of our system architecture. At a high level, the process for passing a μCAPTCHA is the following:

[q] http://josscrowcroft.com/demos/motioncaptcha/

1. The system generates a challenge: the image of a math expression.
2. The user draws the math expression represented in the challenge.
3. The sequence of handwritten strokes are submitted to an online math recognition system.
4. For a challenge to be passed, the ID of the recognition result must match the ID of the mathematical expression presented to the user.

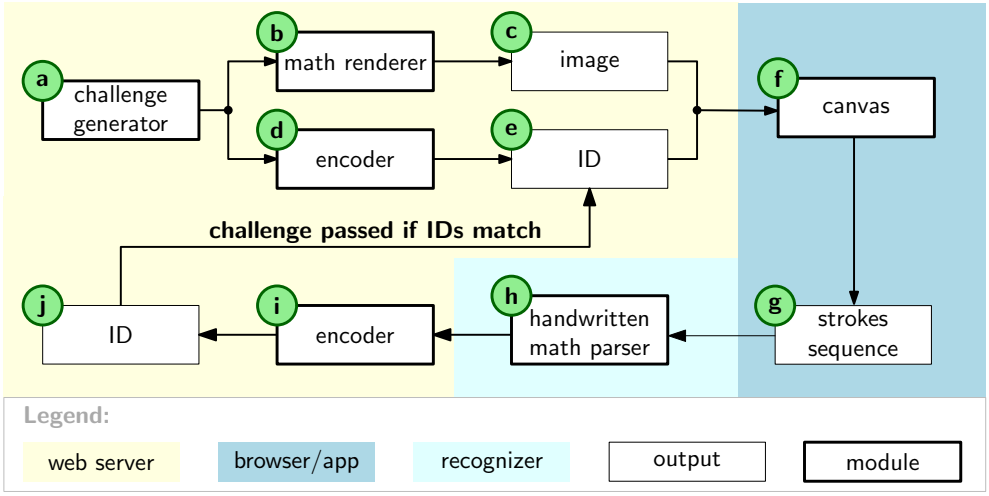


Figure 8.4: Requesting a μCAPTCHA generates a math expression in TEX format (a). The expression is encoded (d, e) and rendered as an image (b, c). The user must draw (f) the presented math expression, generating thus a sequence of strokes (g) that is submitted to a handwritten math expression parser (h). The output of the recognizer is a math expression in TEX format, which is encoded (i) in the same way as (d). The challenge is solved if the final ID (j) matches the challenge ID (e).

A μCAPTCHA ID is a one-way "salted" hash of a TEX equation generated by a PCFG. The server-side encoder knows how to de-salt the hash, which depends on the session and a number of additional factors such as the website requesting the challenge or the checksum of the non-distorted challenge image. So, even if a malicious attacker could mimic our encoder algorithm, these measures would deter a potential attack.

Generation and Recognition of Math Expressions

In order to automatically generate challenges for μCAPTCHA, we manually defined a PCFG that produces math expressions in TEX format. Fig. 8.5 shows some μCAPTCHA examples, where images have been obfuscated.

Special consideration has been put into making our challenges reasonable for humans to solve. On the one hand, successful captchas rely on segmentation problems, as they are computationally expensive [Simard et al., 2003]. μCAPTCHA images use black and white foreground arcs, since these are easily recognized for humans and yet remain difficult for computers to distinguish [Chellapilla et al., 2005]. On the other hand, although PCFGs are defined at the symbol level [Zanibbi and Blostein, 2012], our PCFG for challenge generation is defined at the stroke level because in μCAPTCHA the user has to enter strokes. Thus, a "longer" challenge means that the user has to write more strokes, not necessarily that the expression has more symbols. For instance, $\boxed{2-3}$ (3 symbols, 3 strokes) is much faster to write than $\boxed{F \neq \pi}$ (3 symbols, 9 strokes).

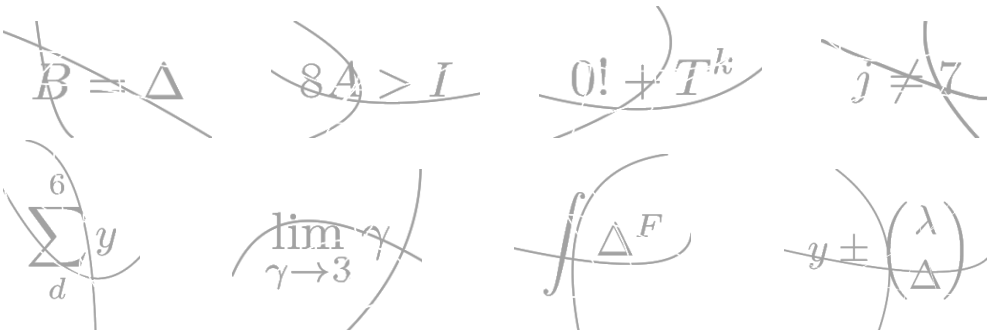


Figure 8.5: Examples using between 4 and 9 strokes per mathematical expression.

There are 3 different sources of error in μCAPTCHA:

1. The challenge is too hard or unclear, which may be skipped.
2. The user draws an incorrect symbol.
3. The math recognizer does not accept the answer.

The latter is worth of consideration, as it introduces some degree of indeterminism due to recognizing hand-made input. Therefore, it is important for the math recognizer to reach high accuracy results.

μCAPTCHA is built upon `seshat`, the open-source recognizer of online handwritten math expressions that implements the approach developed in this thesis. The 2D-PCFG used in the math recognition parser has been obtained from the PCFG we used to generate the challenges.

Initially, we considered the full set of 101 symbols used in CROHME dataset (see Section 7.1.2). Since the number of samples provided for each class is quite unbalanced, we manually labeled new symbols until we obtained 500 samples per class. Then, for each symbol class we used 400 samples for training, 50 samples for validation, and 50 samples to form the test set. As a result, we trained the BLSTM-RNN symbol classifiers with 40,400 symbols. The error rate of isolated symbol recognition was 6.97%. This was not sufficient for μCAPTCHA, so we removed the symbols that caused most of the errors according to the classifier's confusion matrix. After this, the error rate was as low as 0.88% considering 66 symbols. Fig. 8.6 shows the symbol sets before and after removal of conflicting classes.

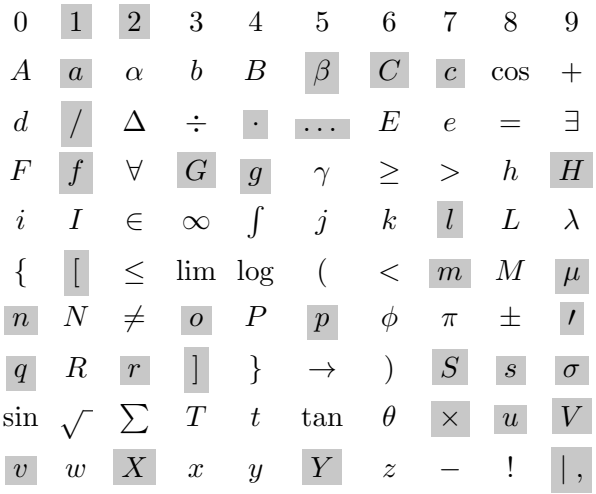


Figure 8.6: Original set (101 symbols) and reduced set (66 symbols), by removing those symbols indicated in gray background color.

Reducing the set of math symbols resulted in an isolated symbol classification error of near zero, but the structure of the math expressions has to be recognized as well. Thus, aiming at a better system, we limited our PCFG so that structural ambiguities were removed. For instance, we decided that a letter followed by a number can only be subscript or superscript. With these improvements, the error of our math recognition system should be very low.

The error at expression level will be evaluated in next section, because we need expressions accepted by our constrained grammar.

Despite reducing the number of symbols and constraining the grammar, the number of accepted mathematical expressions is very large. Concretely, with a vocabulary of m math symbols and r spatial relations, there are approximately $m^N r^{N-1}$ math expressions of N symbols. μCAPTCHA uses 66 symbols and 5 spatial relations, so a challenge with 3 symbols has about 7M of possible combinations, which represents 22.7 bits of information entropy. This is actually an upper bound, since not all symbols can use all spatial relations. By way of comparison, a 4-digit PIN has 10^4 combinations and 13.2 bits of information entropy.

Web Service

Our captcha proposal provides a JSON-based REST API to become backend-agnostic. This way, developers can use their preferred technology stack to deploy μCAPTCHA on websites or native mobile apps. The API provides two entry points: one for requesting a challenge and other for solving the challenge (Fig. 8.7). Both API entry points require a registered user token to be submitted on each request. Otherwise the server responds with a "403 Forbidden" HTTP status code, which indicates that the server can be reached and understood the request, but refuses to take any further action. When a challenge is requested, the developer can point to PNG and MP3 files by concatenating the response ID with `.png` or `.mp3` extensions. To solve the challenge, the user must submit a sequence of online strokes in the following format:

$$
\begin{array}{l}
[\\
\quad [\,[x_1, y_1, t_1], \ldots, [x_N, y_N, t_N]\,], \qquad \texttt{//First stroke} \\
\quad \ldots, \\
\quad [\,[x_1, y_1, t_1], \ldots, [x_M, y_M, t_M]\,] \qquad \texttt{//Last stroke} \\
]
\end{array}
$$

where x and y are coordinates and t is their timestamp.

A `status` code informs the application interfacing with our web service whether the result was successfully processed (0: no error) or not (code > 0 otherwise). A challenge is passed when the value of `msg` equals `"success"`. All passed challenges are periodically fed back to our math recognizer so that it can learn different writing styles and improve accuracy.

Request	GET http://url/user_token/challenge
Response	data:{ "status":0, "id":"http://url/hash" }

Request	POST http://url/user_token/solve/id data:[strokes array]
Response	data:{ "status":0, "msg":"success" }

Figure 8.7: REST API. We have developed an accompanying web-based prototype that interfaces with our web service.

Web Application

We have developed a web-based prototype that interfaces with our web service. In order to save valuable touchscreen space, we decided to put the challenge as a background image in a web canvas, as shown in Fig. 8.3. However, a developer implementing other μCAPTCHA interfaces may decide to present the user with the image and the canvas separately (Fig. 8.8), as the user is actually not required to trace the symbols of the math expression. It is required, however, to preserve the spatial relations between symbols.

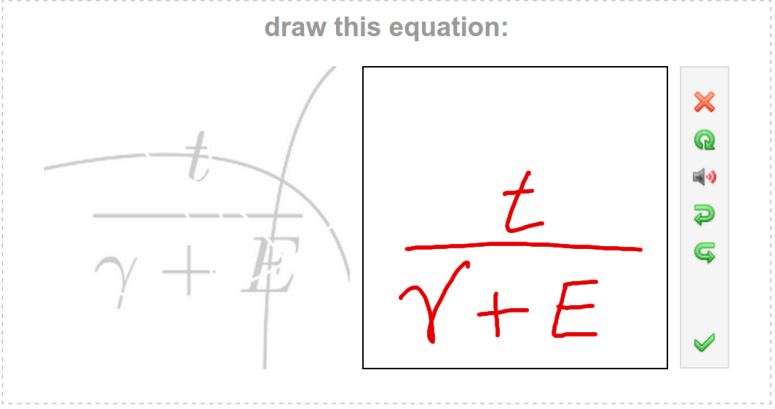


Figure 8.8: Alternate UI design. Eventually we opted for the compact version (Fig. 8.3) in order to save screen space.

Similar to other captchas, to account for challenges that are unreadable or too difficult to write, our application allows users to request a new challenge. Also, when a μCAPTCHA is wrongly solved, the user is presented with a different challenge; there is no option to retry the same μCAPTCHA once submitted. Further, in order to provide an accessible option for all users, our application features the possibility of hearing the mathematical expression and

letting the users to write it down in plain text. In this case, the web canvas is replaced by a regular text field. Under the hood, the audio synthesis is performed with Festival,[r] an open source speech synthesis system. English and Spanish voices are currently available, as they are shipped with the latest Festival version.

8.3.4 Evaluation

In this section, we performed two studies in order to investigate the possibilities of μCAPTCHA. The first one analyzed the strengths of μCAPTCHA against printed math expression recognition systems. The second one was an in-lab user study, aimed at comparing μCAPTCHA against two alternatives.

Attacking μCAPTCHA

We simulated a fundamental attack consisting in scanning μCAPTCHA challenges offline with math OCR software. We generated thousands of challenges and tried to automatically solve them with InftyReader,[s] a very competitive system for recognizing printed math expressions.

μCAPTCHA images are rendered in lightgray color, which caused InftyReader to misrecognize all expressions. Images were thus binarized and submitted again to InftyReader, but it could not recognize any expression because of the foreground noise. Therefore, we sought a more sophisticated attack.

Aimed at removing the foreground noise while preserving the math symbols, we applied erosion and dilation operations with different operator sizes. Fig. 8.9 shows some examples of the images resulting after these morphological operations. Even so, none of these attempts allowed InftyReader to properly recognize any expression because the lines and arcs used as noise have similar width to that of the math symbols, thus either the noise was not successfully removed or parts of the symbols disappeared afterward.

We conclude that μCAPTCHA cannot be broken with out-of-the-box math OCR or basic image preprocessing techniques. This is because offline recognition usually relies on connected components analysis, which in our case is obfuscated by using foreground noise as lines and arcs. Even in the case that noise could be removed with a sophisticated preprocessing technique, at least two further issues should be addressed to break a μCAPTCHA. First, the denoised image should be scanned with a competent math OCR. Then, it

[r] http://festvox.org
[s] http://www.inftyproject.org

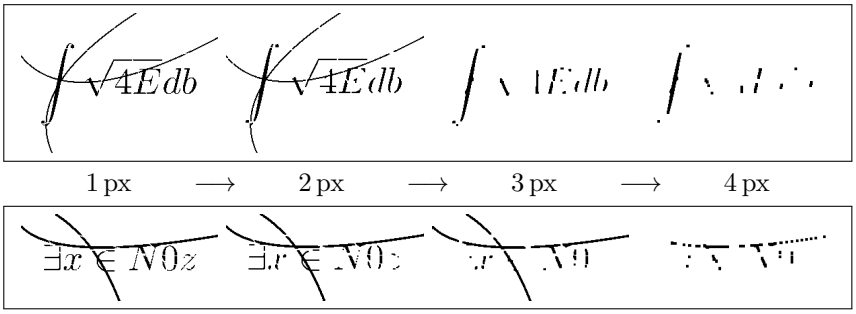


Figure 8.9: Attacking μCAPTCHA. Noise reduction examples use morphological operators (erosion plus dilation) of variable size.

still should be required to generate the online sequence of strokes that represents the math expression. As a two-dimensional problem, not only must the symbols be correct, but also properly arranged.

User Study

This study was aimed at comparing μCAPTCHA with reCAPTCHA (currently the most popular captcha system) and MotionCAPTCHA (a similar alternative to ours, from a user interaction's perspective). Our hypothesis was that drawing is better than typing on a mobile device in several ways.

Prior to this user study, we carried out a preliminary pilot experiment in order to examine the actual performance of our math recognizer. Also, because we needed to collect some samples of expressions for tuning the parameters of the recognition system. As we used the same PCFG to generate challenges and recognize expressions, we can directly compare the TEX outputs and tune the parameters such that expression recognition rate is maximized.

We advertised the study widely to get a sample of participants with diverse backgrounds. Eventually we recruited 10 participants (3 females) aged 25–35 (M = 30.4, SD = 3.4). Two of them were left-handed. All participants had submitted a text-based captcha at some time while browsing the web, though nobody had technical background in computer security or captcha design. Each participant was given a gift voucher of $10 at the end of the study.

The experiment was a within-subjects repeated measures design with three conditions: reCAPTCHA (Fig. 8.2), MotionCAPTCHA (Fig. 8.10), and μCAPTCHA. We measured error rates and three time-related measures: processing time (time between showing up the captcha and start entering the solution), execution time (time spent solving the captcha), and overall solving

time (until clicking on the submit button).

All challenges were solved on a Nexus 4 smartphone (Android 4.4.4) with the Chrome mobile browser 36. μCAPTCHA challenges were set to randomly range between 3 and 6 strokes per expression, which is approximately 2–6 symbols per expression. In addition, some adjustments were made to the other captcha systems:

- reCAPTCHA currently displays Google street view images by default. Then, after solving exactly 5 challenges they become the usual two words (a verification word, to which reCAPTCHA knows the answer, and an unknown word which comes from an old book). We ensured that the two-word version was always shown to the participants, so that everyone tested it under the same settings.

- MotionCAPTCHA is an insecure option for many reasons. However, it is similar to μCAPTCHA in terms of interaction, c.f. drawing instead of typing. Therefore, we instrumented MotionCAPTCHA to reach the level of its peers. The prototype uses the same interface as μCAPTCHA, the same foreground noise technique, and a similar hashing function to encode the challenges. In addition, 3 unistroke gestures are always shown to the user, which increases the challenge combinations from 16 to 16^3. The server-side recognizer is a nearest-neighbor classifier with Euclidean distance [Wobbrock et al., 2007].

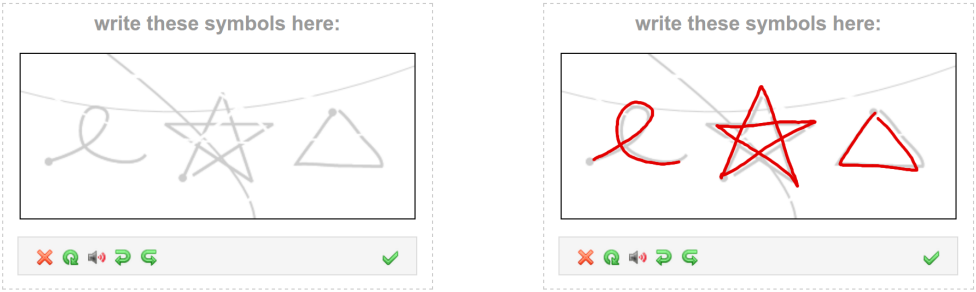


Figure 8.10: MotionCAPTCHA example, using our UI's look and feel.

Participants were briefly introduced the captcha systems they would use and could test each for at most 5 minutes. They had to hold the smartphone in portrait position and were told to go at their normal working pace. They had to solve 25 challenges with each system; resulting thus in 250 observations per condition. Participants were informed whether they successfully solved a

challenge or not as they went, as it is a better simulation of real-world use cases. All systems automatically advanced to the next challenge, regardless it was correctly solved or not. Conditions were counterbalanced and presented to the participants randomly, in order to reduce the chance of learning effects.

With a view to get a better picture of the study, participants filled in the SUS and NASA-TLX questionnaires on a nearby computer. They were also asked to score each captcha system in a 5-point Likert scale in terms of: usefulness, ease of execution, ease of understanding, and ease of learning.

Analysis of Recognition Accuracy As shown in Table 8.1, μCAPTCHA was found to be more accurate than reCAPTCHA and MotionCAPTCHA. The difference in recognition accuracy between reCAPTCHA and μCAPTCHA was not significant, whereas MotionCAPTCHA performed significantly worse than its peers. We conclude that μCAPTCHA is accurate enough for production use. The bad performance of MotionCAPTCHA is explained by the fact that a single challenge entails 3 unistroke gesture recognition tests, so if just one of these tests is unsuccessful, the challenge is not passed.

Table 8.1: Recognition accuracy results.

System	Accuracy (%)	95% CI*
reCAPTCHA	88.4	[83.84 – 91.80]
MotionCAPTCHA	65.6	[59.52 – 71.21]
μCAPTCHA	90.8	[86.57 – 93.79]

* Wilson interval estimation for binomial distributions.

Analysis of Solving Time Overall, participants spent considerably more time solving a reCAPTCHA than a MotionCAPTCHA or a μCAPTCHA. Results of Table 8.2 revealed that reCAPTCHA was significantly slower than its peers. μCAPTCHA was about one second slower than MotionCAPTCHA, however this was unsurprising because μCAPTCHA challenges had almost twice the number of strokes.

Similarly, execution time results showed that participants spent considerably more time typing a reCAPTCHA than drawing a MotionCAPTCHA or a μCAPTCHA. Users spent a significantly high amount of time typing as compared to drawing. As expected, drawing a μCAPTCHA was slower than MotionCAPTCHA, since μCAPTCHA challenges require entering more strokes.

Table 8.2: Mean solving time. SDs are denoted in parentheses.

Time (s)	reCAPTCHA	MotionCAPTCHA	μCAPTCHA
Processing	5.0 (2.9)	0.9 (0.4)	1.3 (0.7)
Execution	10.2 (3.6)	4.4 (1.2)	5.1 (1.9)
Overall	15.2 (4.5)	5.3 (1.5)	6.5 (2.1)

We observed that reCAPTCHA takes up to one third of the total time to process the challenge, i.e. the user first reads the challenge and then focuses on the text field to start typing. In contrast, participants were considerably faster with the other systems. MotionCAPTCHA and μCAPTCHA required similar solving time, which suggests that our math challenges are both easy to read and understand.

Analysis of Usability and Workload In terms of SUS, reCAPTCHA scored lower than its peers (higher is better), as shown in Table 8.3. Results revealed that reCAPTCHA was found to be significantly less usable than the other systems, while MotionCAPTCHA and μCAPTCHA performed similar.

Table 8.3: Mean usability (SUS) and workload (TLX) scores. SDs are denoted in parentheses. Higher SUS is better, lower TLX is better.

Score	reCAPTCHA	MotionCAPTCHA	μCAPTCHA
SUS	49.7 (16.1)	77.0 (11.5)	82.0 (8.8)
TLX	6.1 (1.9)	4.0 (1.3)	3.9 (1.2)

In terms of TLX, reCAPTCHA scored higher than its peers (lower is better). Results revealed that reCAPTCHA was found to be more cognitively demanding than the other systems. MotionCAPTCHA and μCAPTCHA performed similar.

Analysis of Perceived Utility Our participants found reCAPTCHA to be less likable than the other systems, though they acknowledged that it pursues the considerate goal of digitizing books. This is why reCAPTCHA scored higher in terms of usefulness but was around a neutral score for the rest of the assessed statements (see Table 8.4). Overall, reCAPTCHA was perceived to be significantly less valuable than its peers. It was interesting to note that

MotionCAPTCHA, despite its low accuracy, was rewarded by the participants as being statistically similar to μCAPTCHA. This suggests that mobile users are eager to try captchas that are best suited to mobile devices.

Table 8.4: Mean subjectivity scores (higher is better). SDs are denoted in parentheses.

Perception	reCAPTCHA	MotionCAPTCHA	μCAPTCHA
Usefulness	4.0 (0.8)	3.8 (0.7)	4.5 (0.5)
Execution	2.9 (1.3)	4.0 (0.8)	4.4 (0.5)
Understandability	3.2 (1.3)	4.2 (0.7)	4.3 (0.5)
Learnability	3.2 (1.2)	4.3 (0.7)	4.3 (0.5)
Overall	3.2 (1.2)	4.2 (0.7)	4.3 (0.5)

Qualitative Observations Participants were concerned about the fact that unistroke gestures must be performed in a unique way: *"even though it is very easy to reproduce the gestures, the recognizer fails too often."* In contrast, in μCAPTCHA it is possible to write math symbols in different ways, which provides the user with more flexibility. In this regard, one participant stated: *"I'm not very good at writing... I'm surprised how accurate is the math recognizer!"* Finally, other participant made an interesting observation: *"Initially I considered* μCAPTCHA *to be more complex than MotionCAPTCHA, but then I realized that you are just drawing what you see in the background."*

Labeled Dataset As a result of the challenges solved by our participants, 2,787 online handwritten math expressions have been automatically annotated. The math recognition system outputs a TEX transcription together with an InkML file describing the recognition result. Therefore, when a challenge is solved we obtain a valid sample that is annotated at the stroke level with symbol segmentations, symbol classes, and the structure of the expression. μCAPTCHA is thus channeling the user effort into building valuable machine learning datasets. For instance, the data can be useful for training handwritten math expression recognition systems, for which the annotation process is usually tedious and time-consuming [MacLean et al., 2011].

Follow-up Study: Improving Recognition Accuracy

Overall, the accuracy rate of μCAPTCHA is competitive enough for production use, however we wondered if it could be further improved. One way to do so is by improving the math recognizer. Nevertheless, this is a non-trivial task that requires expert knowledge in pattern recognition or machine learning procedures. A more plausible option consists in introducing a confidence measure, so that the outcome of the recognizer is not as hard as a true/false binary decision. In fact, reCAPTCHA intentionally tolerates some errors depending on how much they trust the user giving the solution. This is in line with a question raised by one of our participants: *"most of the time I did not understand why reCAPTCHA accepted my answer because I had to do a very big guess on one of the words."* Eventually we adopted the following solution.

For mathematical expressions, EMERS [Sain et al., 2010] is a well-defined tree edit distance for performance evaluation (see Section 6.2.2). EMERS is not a normalized distance, but it calculates the set of edit operations to transform a tree into another such that if both trees are identical EMERS is equal to zero. Fig. 8.11 shows the gain achieved in accuracy incurred by different EMERS thresholds.

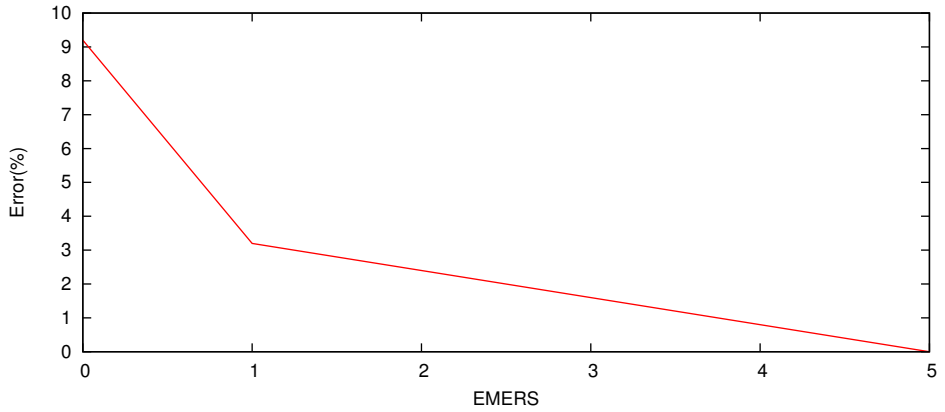


Figure 8.11: Improving accuracy by using different EMERS thresholds.

As observed, EMERS-1 (one tree-edit distance) leads to 50% of improvement in current recognition rates (Fig. 8.11). However, this comes at the cost of increasing the chance of false positives. Thus, it is important to seek a balance between using a binary decision as recognition result (hard constraint) or being a bit fault-tolerant in order to improve the user's acceptance. To this end, we have decided to use EMERS-1 in case the recognition result does

not match the challenge submitted to the user. This should increase the user experience without impacting security.

Regarding the automatic annotation of math expressions, only those expressions that have been perfectly recognized (EMERS-0) are actually saved as ground truth data.

8.3.5 Discussion

Bursztein et al. [2014] analyzed the interactions of visual features used in today's captchas, in order to understand how they affect captcha's difficulty and user perception. Unexpectedly, it was found that accuracy and solving time are not good predictors of user preference. Instead, it is more effective to use the captcha as a medium for engagement with the user, and examine the interaction holistically. This explains why our participants scored MotionCAPTCHA and μCAPTCHA similarly; see Table 8.3 and Table 8.4. However, our evaluation tasks were artificial and therefore we suspect that in a real-world situation this perception toward MotionCAPTCHA might change due to its relatively low accuracy.

HIPs, and by extension captchas, are based on open problems in artificial intelligence (AI) that induce security researchers, as well as otherwise malicious programmers, to work on advancing the field of AI. HIPs are thus a win-win situation [von Ahn et al., 2004]: either it is not broken and there is a way to differentiate humans from computers, or it is broken and an AI problem is solved. While μCAPTCHA does not guarantee that is secure against highly sophisticated attacks, it is not trivially breakable. However, like every captcha system, μCAPTCHA could be defeated by human manpower. For example, spammers could pay a developer to aggregate our challenges and feed them one by one to a human operator.

Our current implementation of μCAPTCHA has a number of limitations that are intended to be addressed in future work:

- It would be desirable to incorporate more math symbols, in order to further increase μCAPTCHA's information entropy. However, our preliminary studies suggest that doing so may increase the error rate of the recognizer. A plausible option would be balancing EMERS and expression length; e.g. EMERS-2 would not be tolerable for use with expressions comprising 3 symbols, but it may be adequate for use with expressions having 6 symbols. Also, the set of symbols was constrained by the database of math expressions. A larger dataset of handwritten math symbols would provide more symbols to train the math parser.

- The lines and arcs of the foreground noise are drawn in an uncontrolled fashion. It may happen that they occlude parts of the expression, such as a minus sign, which could lead the user to enter a wrong solution to the challenge presented. In fact, one of our participants complained about this issue. In any case, if the user cannot distinguish what symbols are being shown, it is possible to reject the challenge by requesting another one. We log all unsolved challenges for later analysis.

- It may be the case that a mathematical expression does not make sense; e.g. $\boxed{2x\ \forall y}$. This may confuse users with advanced math knowledge, though we have not received such a complain. We believe that a mechanism to generate "mathematically correct" expressions would be desirable.

On another line, it is worth noting that our web service can be integrated in native mobile apps; however we expect μCAPTCHA to be largely used on websites. This was the main reason why we developed a web-based application to interface with our web service. In this context, we should mention that the application is tailored to HTML5 browsers. Therefore, drawing on a web canvas might not work in old devices or old browsers. Also, drawing does not work at all in browsers that have disabled JavaScript.

Lastly, if μCAPTCHA were largely used, it would create a vast dataset of labeled online handwritten math expressions. This resource would help the advancement in math research, which is lately bringing more attention to students as a result of the rapid growth of mobile devices. For instance, education is moving toward computer-based solutions at the expense of traditional (paper-based) books, so handwritten input is likely to play an important role in the classrooms in the near future. Other applications are introduced in Section 1.1, for which μCAPTCHA would contribute in creating worthwhile resources.

8.4 Layout Analysis of Structured Documents

In this thesis we developed an approach for math expression recognition based on parsing 2D-PCFG. The two-dimensional extension of PCFG studied in this document is a powerful formalism that can be applied to other problems. Therefore, we explored the application of 2D-PCFG to layout analysis of structured documents. Probabilistic grammars are suitable for this problem because they can account for the syntactic structure of the document and the two-dimensional dependencies between regions.

8.4.1 Introduction

Page segmentation is a fundamental problem of Document Image Analysis (DIA) which is important for subsequent document analysis and recognition tasks. Document image segmentation intends to detect homogeneous relevant zones in a given document and finding out the structural relations among these zones [Shafait et al., 2008]. The relevant zones in DIA depend on the task and they can be drawings, textual zones, special symbols, etc.

Many successful image segmentation techniques have been defined in the past for typeset documents [Shafait et al., 2008]. Several contests have been held for this type of documents where a common framework is defined in order to be able to compare existing techniques [Antonacopoulos et al., 2013a,b]. Developing generic image segmentation techniques for historical handwritten documents is a very difficult task due to the absence of general editing rules in the past, since the editing rules were usually different for each collection.

Some historical handwritten documents exhibit regularities similar to typeset documents, and image segmentation techniques used for typeset documents can be considered for historical handwritten documents [Antonacopoulos et al., 2011]. Segmentation of this kind of documents has been approached in the past with geometrical techniques. In [Bulacu et al., 2007] projection profiles were mainly used for page layout analysis of documents with very satisfactory results. But for many other documents, page segmentation techniques that rely on explicit isolation of elements like characters, words or lines are often not useful. For those documents, integrated approaches seem more appropriate. We focus on determining the structure and the segmentation of textual zones in images of this second type of historical handwritten documents, concretely in marriage license books [Romero et al., 2013] (see Figure 8.12). This step is crucial for subsequent text recognition processes like line detection and extraction [Likforman-Sulem et al., 2007] and later transcription, or word spotting [Toselli et al., 2004b; Puigcerver et al., 2014].

Probabilistic Graphical Models (PGM) offer a natural framework to tackle these segmentation problems and to relate segmented units represented here as random variables, since it easily allows to represent dependencies between them [Jordan et al., 1999; Wainwright and Jordan, 2008; Koller and Friedman, 2009]. However, computing exact inference on these models may be challenging depending on the structure that they present. In this case we must resort to other approximate methods like the Graph Cut algorithm [Boykov et al., 2001] or some variations of the Belief Propagation (BP) algorithm [Yedidia et al., 2003]. Within this formal framework, Cruz and Terrades [2012] proposed a solution for classifying the different textual zones that are present in marriage license books, although no structure detection was performed. In that research, pixel classification based on texture features obtained from the Gabor transform are compared with Relative Location Features [Gould et al., 2008]. Both types of features are combined in a Conditional Random Field [Lafferty et al., 2001] to take into account contextual information in the classification process of the pixels.

In order to address both the detection of textual zones and the analysis of structural relationships among these zones, we consider the use of structural models. PCFG are a powerful formalism of syntactic pattern recognition which has been used previously for DIA [Jain et al., 2001; Handley et al., 2005]. 2D-PCFG is a well known formalism that has been studied in the past for two-dimensional parsing [Crespi Reghizzi and Pradella, 2008]. As we have seen in the approach for math recognition developed in this thesis, this type of grammars are able to represent efficiently contextual two-dimensional relations that are important for page segmentation.

In this research we study the application to segmentation of structured documents of the approach based on parsing 2D-PCFG developed in this thesis for math expression recognition. We propose a formal model that integrates several probability distributions for textual zone segmentation and structural analysis directly into the parsing process of 2D-PCFG. We deal with the estimation of the probabilistic sources and we compare our approach to solutions based on PGMs.

8.4.2 Segmentation of Structured Documents

Marriage license books are documents that were used for centuries to register marriages in ecclesiastical institutions. Each marriage is represented by a record and the transcription of these documents has been considered very interesting for demography and migratory research [Esteve et al., 2009]. Each unit of information is composed of several related textual regions. Two relevant

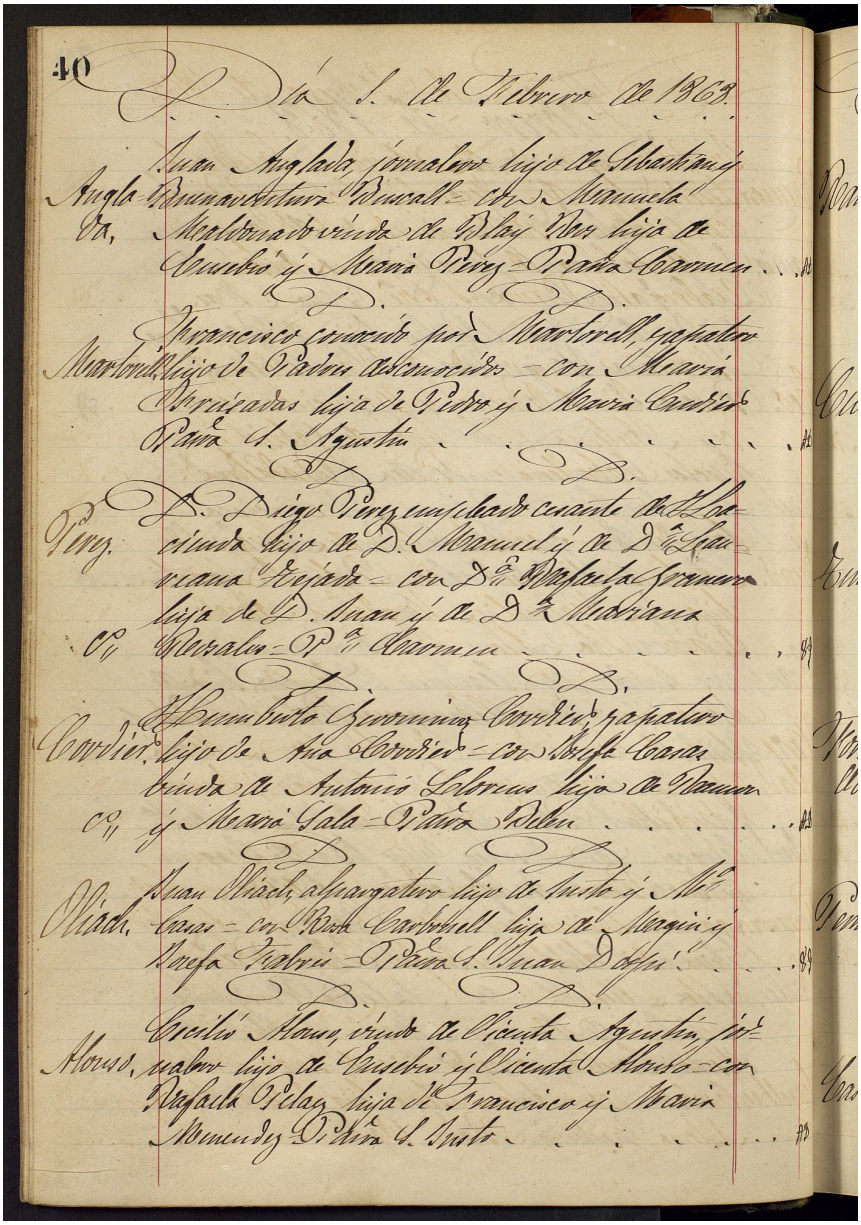


Figure 8.12: Example of page of a marriage license book containing six records.

page segmentation problems can be stated for these documents. First, to segment and classify the different textual units of the records. And second, to find out the syntactic structure of the records in a given page.

Most of these books have a structure similar to an accounting book. Figure 8.12 shows an example of page of a marriage license book belonging to a collection of 291 books conserved at the Cathedral of Barcelona. The pages in these books were orderly written, and although there are differences over the centuries, the layout in each page was quite rigid. Every book is divided in two parts: the first part is an index of surnames and the second part contains the marriage license records (see [Romero et al., 2013] for a more detailed description of this collection). We focused on the segmentation of the pages in the second part of the book.

Each page contains several records, such that each one is associated with a marriage license. Each record has in turn a *husband surname's block* (Figure 8.13.a), the *main block* (Figure 8.13.b), and the *tax block* (Figure 8.13.c). Note that the documents can have additional textual zones, like the date that can be seen at the beginning of the page (it can also appear in the middle of a page), and the two large calligraphic letters[t] that separate the consecutive records that were registered the same day. These additional zones were ignored in this study, i.e. they are considered background because they were not relevant for subsequent transcription tasks. The process for creating the ground-truth requires marking the minimum rectangle containing the identified classes: *Body*, *Name* and *Tax*. All the pixels that did not belong to any of these regions were considered background.

The final goal in those documents is to obtain the transcription of each marriage license. The problem is to correctly isolate every record in a page, and to relate their corresponding parts, that is, the surname, the body text and the tax associated with each entry. We aim at detecting the bounding boxes around the main parts of each record. Note, that a fine-grained detection of the frontiers of each zone would be ideal, but this is difficult because sometimes two zones overlap if rectangular bounding boxes are used as in this approach (see the lower record in Fig. 8.13).

The problem of detecting the records can be stated as two different problems: first, to classify the textual zones into the previously mentioned classes (*Background*, *Name*, *Body* and *Tax*); and second, to detect the complete set of records of each page. To address this problem, first we review related work based on PGMs to solve the segmentation of images of documents. These

[t]These letters are D. D. that is the abbreviation of "Dit dia" which means "The mentioned day" in Catalan.

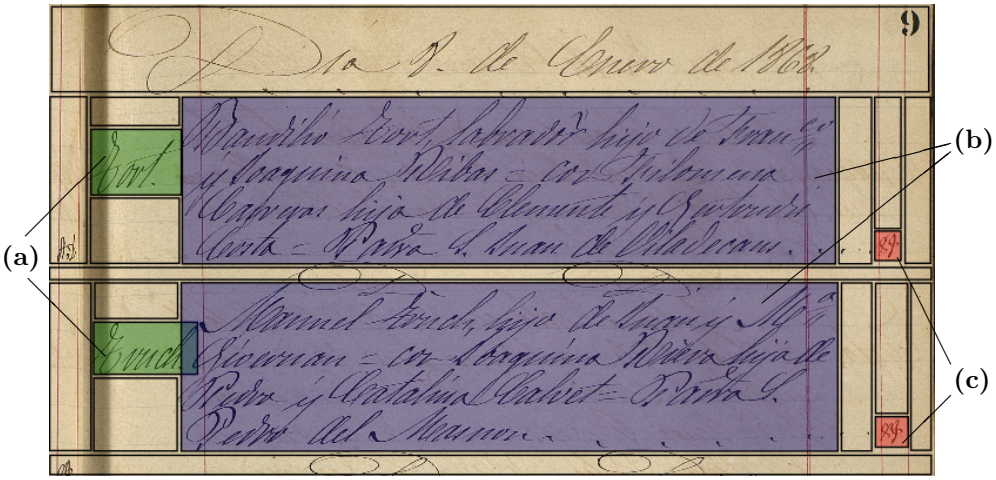


Figure 8.13: Example of the page segmentation problem for two records. Several background zones are considered and each record is composed of three parts: (a) Name (b) Body (c) Tax.

graphical models become our baseline approach. Second, we present a model based on 2D-PCFG that solves the segmentation of the full document using structural and probabilistic information. Finally, we describe two sets of text classification features used to classify the image regions according to the graphical information, and the evaluation performed on this corpus.

8.4.3 Probabilistic Graphical Models

When we tackle the problem of image labeling, PGMs [Koller and Friedman, 2009] provide a proper framework to represent the structure and the relationships between the variables in the model. In this representation the different variables are distributed in a graph structure, where it can be depicted as a directed or undirected graph depending on the type of dependencies represented. This graph is composed by a set of nodes representing the different set of variables, and a set of edges denoting the dependencies between the nodes.

In the case of image labeling, a natural way for representing the dependencies between the pixels of the image is by means of a two-dimensional grid-like structure, which can be modeled by a Markov Random Field (MRF) [Li, 1995]. In this representation each pixel in the image is represented by a node in the graph, although in some tasks it is also common that a node represents a group of pixels clustered in cells or superpixels [Gould et al., 2008].

In this problem the objective is to compute *Maximum a Posteriori* (MAP) probability to find the combination of class labels c for each pixel in x that maximizes the PGM probability. One way to model this distribution is in terms of the energy associated with a Conditional Random Field (CRF) [Lafferty et al., 2001], conditioning the probability with respect to a set of computed features:

$$p(c \mid x) = \frac{1}{Z(x)} \exp\left\{ \sum_i \psi(x_i, c_i) + \sum_{(i,j)\in\varepsilon} \phi(c_i, c_j) \right\} \tag{8.4.1}$$

where $\psi(x_i, c_i)$ represents the local potentials at each pixel i, and $\phi(c_i, c_j)$ the pairwise potentials of assigning the labels c_i, c_j to the neighbor pixels i, j. The constant $Z(x)$ represents the partition function, a normalization factor to ensure the proper definition of the probability distribution. In some types of structures, as in the case of grid-like graphs, a large amount of variables may result in the impossibility of providing an exact computation of this value, leading to the need of using approximate methods to achieve this task [Roth, 1996].

Many methods have been proposed to perform inference in PGMs, that is, to obtain the likelihood or the conditional probability with a model for a given input. However, the problem of computing exact inference in grid-structured CRFs is known to be a NP-hard problem and becomes intractable when we model a large number of variables [Cooper, 1990]. Nevertheless, there are many methods in the literature that provide approximate solutions to the problem. One example is the Graph Cut algorithm [Boykov et al., 2001] used in many segmentation tasks [Kumar et al., 2007]. The method relies on the fact that many computer vision problems can be formulated in terms of an energy minimization function, and provides a local minimum based on the most likely cut in the graph. Another family of methods used to perform approximate inference in graphical models are the sum-product message passing algorithms. Within this type of algorithms are the Belief Propagation (BP) algorithm [Yedidia et al., 2003], and the version for loops Loopy Belief Propagation (LBP) [Weiss, 2000], that is able to perform inference in the case of grid-structured CRFs. There are also other algorithms that follow different approaches. One example is the Iterated Conditional Models (ICM) [Besag, 1986], an algorithm for optimization that follows a search paradigm. In this study we use the three algorithms stated before for the inference in PGMs.

8.4.4 Grammatical Model

We studied the application of parsing 2D-PCFG in order to compute the most likely structure and segmentation of a document. This powerful model intends to tackle the logical layout problem in combination with text classification features. A context-free model is a natural way to account for both the horizontal and vertical context of the problem, where there are dependencies among rows, columns and 2D regions.

The formal definition of 2D-PCFG is provided in Section 2.4.1. In this problem, the entire document must be parser, thus we define 2D-PCFG that are able to deal with two-dimensional matrices. There are only two spatial relationships ($r \in \{\mathrm{H}, \mathrm{V}\}$) in this two-dimensional extension: horizontal concatenation (H) or vertical concatenation (V). Given a rule $A \xrightarrow{r} B\ C$, the combined subproblems B and C must be arranged according to the spatial relation constraint, i.e. horizontally adjacent and same height for $r = \mathrm{H}$, and vertically adjacent and same width for $r = \mathrm{V}$. This simple extension does not require a spatial relationship model and is enough to account for the problem we are dealing with. The segmentation of the input document can be obtained as the most likely derivation given a 2D-PCFG, such that the region that defines the input image is recursively divided either vertically or horizontally into smaller rectangular regions.

Parsing Algorithm

Given a page image, the problem is to obtain the most likely parsing according to a 2D-PCFG. For this purpose, the input page is considered as a two-dimensional matrix I with dimensions $w \times h$ and each cell of the matrix can be either a pixel or a cell of $d \times d$ pixels. Then, we define an extension of the well-known CYK algorithm to account for two-dimensional structures. We have basically extended the algorithm described in [Crespi Reghizzi and Pradella, 2008] to include the probabilistic information of our model, and the integration of different probabilistic sources studied in this thesis.

The CYK algorithm for 2D-PCFG is essentially a *dynamic programming* method, which fills in a parsing table $\mathcal{T}$. Following a notation very similar to [Goodman, 1999], each element of $\mathcal{T}$ is a probabilistic nonterminal vector, where their components are defined as:

$$\mathcal{T}_{(x,y),(x+1,y+1)}[A] = \hat{p}(A \Rightarrow z_{(x,y),(x+1,y+1)}) \tag{8.4.2}$$

$$\mathcal{T}_{(x,y),(x+i,y+j)}[A] = \hat{p}(A \overset{+}{\Rightarrow} z_{(x,y),(x+i,y+j)}) \tag{8.4.3}$$

Each region $z_{(x,y),(x+i,y+j)}$ is defined as a rectangle delimited by its top-left corner (x, y) and its bottom-right corner $(x + i, y + j)$. We denote

$$\ell_{i\times j} = \ell(z_{(x,y),(x+i,y+j)})$$

as the size $(i \times j)$ of the subproblem associated with a region $z_{(x,y),(x+i,y+j)}$. The probabilities $\hat{p}$ represent the probability of the most likely derivation from nonterminal A resulting in the region z.

If the size of the subproblem is larger than 1×1, then there exists some binary rule ($A \xrightarrow{r} B\ C$, with $B, C \in N$, and $r \in \{H, V\}$) and some split point k such that, in a similar way to [Goodman, 1999], we can divide the problem in two subproblems:

$$\begin{aligned}\hat{p}(A \overset{+}{\Rightarrow} z_{(x,y),(x+i,y+j)}) = p(\ell_{i\times j} \mid A)\ \max_{B,C}\ \{ \\ \max_{1\le k<i} p(A \xrightarrow{H} B\,C)\ \hat{p}(B \overset{+}{\Rightarrow} z_{(x,y),(x+k,y+j)})\ \hat{p}(C \overset{+}{\Rightarrow} z_{(x+k,y),(x+i,y+j)})\ , \\ \max_{1\le k<j} p(A \xrightarrow{V} B\,C)\ \hat{p}(B \overset{+}{\Rightarrow} z_{(x,y),(x+i,y+k)})\ \hat{p}(C \overset{+}{\Rightarrow} z_{(x,y+k),(x+i,y+j)})\ \}\end{aligned} \tag{8.4.4}$$

where a new hypothesis is computed from two smaller subproblems, such that the probability is maximized for every possible vertical and horizontal decomposition resulting in the region $z_{(x,y),(x+i,y+j)}$. It should be noted that the 2D-PCFG provides syntactic and spatial constraints $p(A \xrightarrow{r} B\,C)$, and we have also included the probability $p(\ell_{i\times j} \mid A)$ that a nonterminal A accounts for a problem of size $i \times j$.

The probability $p(\ell_{i\times j} \mid A)$ has two effects on the parsing process. First, it helps to model the spatial relations among every part of a given page because there is a specific nonterminal for each zone of interest. For instance, this can be seen in Figure 8.13 where the size of the background region on top of the page will be different from the size of the background zone over a tax region. Furthermore, many unlikely hypotheses are pruned during the parsing process due to its size information, hence, it speeds up the algorithm.

Considering the definition of the matrix parsing table $\mathcal{T}$ (Eq. (8.4.3)), the expression of the Eq. (8.4.4) can be rewritten to obtain the general term of

the parsing algorithm. Thus, for all i and j, $2 \leq i \leq w,\ 2 \leq j \leq h$, we have:

$$\mathcal{T}_{(x,y),(x+i,y+j)}[A] = p(\ell_{i\times j} \mid A) \max_{B,C} \{$$
$$\max_{1\leq k<i} p(A \xrightarrow{\mathrm{H}} B\,C)\, \mathcal{T}_{(x,y),(x+k,y+j)}[B]\, \mathcal{T}_{(x+k,y),(x+i,y+j)}[C] \ ,$$
$$\max_{1\leq k<j} p(A \xrightarrow{\mathrm{V}} B\,C)\, \mathcal{T}_{(x,y)(x+i,y+k)}[B]\, \mathcal{T}_{(x,y+k),(x+i,y+j)}[C] \ \}$$

For subproblems of size equal to 1×1 and taking into account the definition of Eq. (8.4.2), the derivation probability of a single cell (size region equal to 1×1) can be marginalized according to the class label (terminal) c. Given that we need to calculate the most likely parsing, we can approximate the sum by a maximization, and considering some other usual assumptions the probability of the derivation of a single cell is:

$$\hat{p}(A \Rightarrow z_{(x,y),(x+1,y+1)}) \approx p(\ell_{1\times1} \mid A) \max_{c} p(A \to c)\ p(c \mid z) \qquad (8.4.5)$$

where $p(\ell_{1\times1} \mid A)$ is the probability that nonterminal A derives a subproblem of size 1×1; $p(c \mid z)$ represents the probability that a cell (region) z belongs to class c, and it is described in Section 8.4.5; and $p(A \to c)$ is the probability of a terminal rule for terminal (class) c.

Taking into account the matrix $\mathcal{T}$ (Eq. (8.4.2)), we can rewrite the expression of Eq. (8.4.5) to obtain the initialization term of the parsing algorithm. Thus, for each region z of size 1×1, we have:

$$\mathcal{T}_{(x,y),(x+1,y+1)}[A] \ = \ p(\ell_{1\times1} \mid A) \max_{c} p(A \to c)\, p(c \mid z_{(x,y),(x+1,y+1)}) \quad (8.4.6)$$

Finally, the most likely parsing of the full input page is obtained in element $\mathcal{T}_{(0,0),(w,h)}[S]$ such that S is the start symbol of the 2D-PCFG. It is important to notice that all the probability distributions involved in the parsing process can be learnt from labeled data. The time complexity of the algorithm is $O(w^3h^3|R|)$ and the spatial complexity is $O(w^2h^2)$.

Model Estimation

The model based on 2D-PCFG for parsing structured documents has, in turn, some probabilistic distributions that need to be learnt. First, the probability $p(c \mid z)$ that a certain region z of the image belongs to class c is described in Section 8.4.5. There are two additional distributions that we have to estimate: the probabilities $p(\ell_{i\times j} \mid A)$ and the probability of the rules of the grammar $p(A \to \alpha)$. In order to learn automatically these distributions, we followed

the method for estimation described in 4.4. First, we performed a forced recognition of the training set, and then we estimate the distributions using the parsing results.

Given a certain document, the forced recognition was carried out by providing the probability $p(c \mid z)$ using the ground-truth information. Concretely, for each cell z belonging to class c^* we set $p(c^* \mid z) = 1$ and $p(c \mid z) = 0$, $\forall c \neq c^*$. The remaining distributions were considered equiprobable. In this way, we obtained for each document the best parsing according to the 2D-PCFG model.

On the one hand, the probability distribution of the size for each nonterminal A was estimated according to the occurrences in the forced recognition of the training set as

$$p(\ell_{i\times j} \mid A) = \frac{n(A_{i\times j})}{n(A)}$$

such that $n(A_{i\times j})$ is the number of times that nonterminal A accounts for a region of size $i \times j$ in the training set, and $n(A)$ the total number times that nonterminal A accounted for a region of any size.

On the other hand, the probabilities of the rules of the grammar were estimated using the set of derivation trees obtained from the forced recognition of the training set using the Viterbi score [Ney, 1992] as described in Section 4.4.1.

8.4.5 Text Classification Features

In this section we describe the different features we used for classifying small regions of pixels and how we incorporated them into the 2D-PCFG parsing described above. Following the outline in [Cruz and Terrades, 2012] we used two different set of features, Gabor features as texture descriptors and Relative Location Features [Gould et al., 2008].

Texture features

Gabor transform is a multi-resolution transform commonly used for texture analysis. A bank of filters is defined for several orientations and signal frequencies. Ilonen et al. [2007] proposed a fast implementation of multi-resolution bank of filters in which the only parameters to be given are: the number of orientations (n), the number of resolution levels (m) and the highest frequency (f_{max}). As a result, it is obtained a feature vector g of dimensions $n \times m$, which covers almost all the spectrum of frequencies up to the highest one f_{max}.

The Gabor filter is defined by a sinusoidal wave of complex values modulated by an exponential function [Fogel and Sagi, 1989]. This function is a

Gaussian function centered in the origin of coordinates, with a parameter controlling the size of the function support[u]. In the frequency space, the Gabor filter is also defined by a Gaussian function, centered in the frequency f_0 and support inversely proportional to frequency f_0. Furthermore, in images the filter support has an elliptical shape tuned by three parameters γ, η and θ:

$$\begin{aligned} \psi(x, y; f, \theta) &= \frac{f_0^2}{\pi\gamma\eta} e^{-\left(\frac{f^2}{\gamma^2}x'^2 + \frac{f^2}{\eta^2}y'^2\right)} e^{i2\pi f x'} \\ x' &= x\cos\theta + y\sin\theta, \\ y' &= -x\sin\theta + y\cos\theta \end{aligned} \tag{8.4.7}$$

It is well-known that the Fourier transform of a Gaussian function is again a Gaussian function. In addition, if we scale the support of Gabor filters by a factor of k^{-m}, the support of their Fourier transform are proportional to k^m. In particular, given the definition of Gabor filters in Eq. (8.4.7), the support of Gabor filters in the spatial domain are ellipses with axis proportional to $\frac{\gamma}{f_{max}}k^m$ and $\frac{\eta}{f_{max}}k^m$. The values of η and γ are obtained according to the number of orientations n, the scaling factor k, and the overlapping degree q of filters in the Fourier space as:

$$\gamma = \frac{k-1}{k+1}\frac{\sqrt{-\log q}}{\pi}; \quad \eta = \frac{\sqrt{-\log q}}{\pi\tan\frac{\pi}{2n}}$$

Once we obtained the set of features, we applied a GMM to estimate the probability $p(g \mid c)$ of each possible class c identified in this task: *Background*, *Name*, *Body* and *Tax* . Finally, we defined the probability $p(c \mid z)$ for a cell z for a particular class c required in Eq. (8.4.6) in the case of the 2D-PCFG, and to define the term ψ in Eq. (8.4.1):

$$p(c \mid z) = \frac{1}{|z|}\sum_{g\in z}\frac{p(g \mid c)\,p(c)}{\sum_c p(g \mid c)\,p(c)} \tag{8.4.8}$$

Relative Location Features

Relative Location Features (RLF) were introduced by Gould et al. [2008] as a way to encode inter-class and intra-class spatial relationships as local features. These features are computed from *relative location probability maps* $\mathcal{M}_{c_i|c_j}$, encoding the probability of the class c_i at region i and knowing that at region j the class c_j is found. In other words, $\mathcal{M}_{c_i|c_j}(u_{i,j}) = p(c_i \mid c_j, i, j)$ where by

[u]We refer as support of a Gaussian function the region enclosing 99% of the energy.

$u_{i,j} = (x_i, y_i) - (x_j, y_j)$ we denote the offset between regions i and j, such that (x_i, y_i) and (x_j, y_j) represent the centroid coordinates of regions i and j, respectively. Thus, the *self* and *other* RLF are defined as:

$$v_{c_i}^{other}(i) = \sum_{i \neq j : c_i \neq c_j} \mathcal{M}_{c_i|c_j}(u_{i,j}) p(c_j \mid j)$$
$$v_{c_i}^{self}(i) = \sum_{i \neq j : c_i = c_j} \mathcal{M}_{c_i|c_j}(u_{i,j}) p(c_j \mid j)$$

Moreover, herein $c_j = \arg\max_c p(c \mid j)$ is the class label assigned at region j having the highest probability and $p(c \mid i)$ is the *a posteriori* probability estimated of Eq. (8.4.8). Each of these features, $v_{c_i}(i) = (v_{c_i}^{other}(i), v_{c_i}^{self}(i))$, model the probability of assigning the class label c to a region z taking into account the information provided by the rest of image regions about their position and its initial label predictions. Finally, once we have computed the set of RLF, we are able to compute the probability required in Eq. (8.4.6):

$$p(c \mid z) = w^{app} \log p(c \mid z) + w_c^{other} \log v_c^{other}(z) + w_c^{self} \log v_c^{self}(z) \qquad (8.4.9)$$

where v_c^{other} and v_c^{self} are the different sets of RLF and w^{app}, w_c^{other} and w_c^{self} are the corresponding weights learnt from a logistic regression model. Further details about the process can be seen in [Cruz and Terrades, 2012].

8.4.6 Evaluation

This section describes the experimentation carried out to evaluate the proposed 2D-PCFG model for the task of page segmentation. We compare the results obtained with approaches based on PGMs using three different families of inference methods.

First, we describe the used dataset and the general settings of the experiments performed. Then we describe the general outline from both experiments. Finally, we report several performance metrics and the discussion comparing the different models.

Experimental Settings

The Five Centuries of Marriages (5CofM) dataset is composed of a set of 291 handwritten books including marriage records conducted in the period from the year 1451 until 1905. The set of books includes approximately $550,000$ marriage licences from 250 parishes [Romero et al., 2013]. Despite the great number of volumes in this dataset, currently only a few of them are being

used on different tasks like handwriting recognition, word spotting, or layout analysis. For each of these tasks the corresponding ground-truth was manually obtained by selecting and labeling the different regions on each page, which is a time consuming process. Therefore, due to time limitations we focused on a particular book of the collection for the experiments reported in this study. However, as the documents in all the volumes have the same structure, the proposed model could be applied to the remaining books.

The experimentation is focused on the segmentation of volume 208 of this collection (Figures 8.12 and 8.13 show examples). This volume has 593 pages of which we labeled at pixel level the first 200 pages. We randomly split 150 pages for training, 10 pages as validation set and the remaining 40 for test. The resolution of the images is 300 dpi ($\approx 2750 \times 3940$ pixels).

Following previous work [Cruz and Terrades, 2012] each page of the dataset was divided in cells of 50×50 pixels in order to reduce the computational cost of processing an image at pixel level. In previous studies we tested several configurations of cell sizes and the impact on the results. The experiments using cells of size 25×25 pixels produced lower precision and recall values for all the considered classes in this task. Smaller cell sizes produce that text classification features do not have enough information in order to correctly discriminate among the different classes. Also, cells over 50×50 pixels were not tested since the regions for some of the classes could be smaller than the cell size and they might not be detected.

With respect to the parameters of the texture filter bank, we computed a 36-dimensional feature vector using 9 orientations and 4 frequencies of the filter. These values were chosen to ensure that the Gabor functions cover the frequency space. Additionally, we set the following parameter values: overlapping degree $q = 0.5$, $f_{max} = 0.35$, and the scaling factor $k = \sqrt{2}$. Finally, we also learnt the parameters of the logistic regression used in Eq. (8.4.9) from the training set.

PGMs Experiments

We performed several experiments using the CRF model defined in Eq. (8.4.1) to compute the best label configuration for a page. We used the two sets of features described in previous section to define the local potentials on each cell of the image as described in Eqs. (8.4.8) and (8.4.9). To compute the values for the pairwise potentials we learnt the frequencies for each possible pair of classes from the training set.

We conducted several experiments using different inference algorithms for the CRF. First, we tested the α-β-*swap* version of the Graph Cut algorithm

proposed in [Boykov et al., 2001]. Second, we use the method based in message passing LBP [Weiss, 2000]. Finally we used the ICM algorithm based in the search paradigm [Besag, 1986].

2D-PCFG Experiments

We tested our proposal based on 2D-PCFG to tackle the page segmentation problem on the 5CofM dataset. Given the nature of the problem such that the documents have a known structure, we manually defined the grammar. According to the model described in Section 8.4.4, we had to train several probability distributions. The probabilities of the productions of the grammar and the size probabilities for each nonterminal were estimated from the training data as explained in Section 8.4.4.

The 2D-PCFG model combines probability distributions that were learnt independently, hence, there may be scaling problems when multiplying the different probabilities. For this reason, the resulting probability was obtained such that each distribution had an exponential weight that adjusted the scale of them. As a result, we had to tune three weights: the probabilities of the grammar $p(A \rightarrow \alpha)$, the probability of a region $p(c \mid z)$ and the probabilities $p(\ell_{i \times j} \mid A)$. Then, the weights of the system were tuned using the Downhill Simplex algorithm [Nelder and Mead, 1965] by maximizing the average F-measure for classes *Name*, *Body* and *Tax* when recognizing the validation set.

Results and Discussion

Learning the different models from the training set and using the best parameters of the validation experimentation, we classified the document images of the test set. We assessed the performance of three inference algorithms for PGMs: Graph Cut, LBP and ICM; and a grammar-based approach (2D-PCFG). These models were evaluated in combination with two sets of features: Gabor features and RLF. Table 8.5 shows the results for each class: *Body*, *Name* and *Tax*. The reported metrics are the precision, recall and F-measure at cell level averaged for each page in the test set. Results show that class *Body* was classified with good F-measure rates, whereas classes *Name* and *Tax* represented the most challenging part. This is related with the percentage of the page that represents each region, because it is more difficult to properly classify small regions. Errors are usually made in the boundaries of the regions, therefore, a row or column of cells represents a smaller percentage of class *Body* than class *Name*, and of course than class *Tax* which is usually composed of just a few cells.

Table 8.5: Classification results for different models and text classification features.

Model	Features	Class	Precision	Recall	F-measure
Graph Cut	Gabor	Body	0.91	0.79	0.84
		Name	0.21	0.80	0.32
		Tax	0.50	0.84	0.61
	RLF	Body	0.90	0.92	0.91
		Name	0.66	0.78	0.70
		Tax	0.89	0.43	0.56
LBP	Gabor	Body	0.89	0.83	0.86
		Name	0.27	0.74	0.39
		Tax	0.45	0.79	0.56
	RLF	Body	0.88	0.93	0.90
		Name	0.72	0.71	0.69
		Tax	0.85	0.43	0.55
ICM	Gabor	Body	0.89	0.83	0.85
		Name	0.27	0.74	0.39
		Tax	0.45	0.79	0.56
	RLF	Body	0.89	0.92	0.91
		Name	0.71	0.74	0.72
		Tax	0.90	0.45	0.58
2D-PCFG	Gabor	Body	0.91	0.95	0.93
		Name	0.73	0.86	0.78
		Tax	0.69	0.80	0.71
	RLF	Body	0.90	0.95	0.92
		Name	0.77	0.79	0.77
		Tax	0.78	0.65	0.68

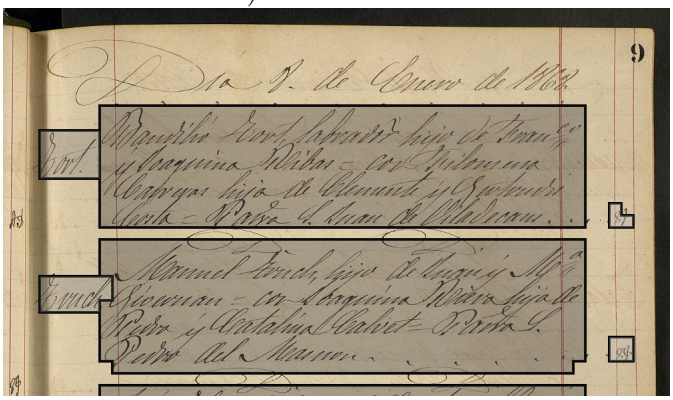
There are two factors to take into account: the text classification features and the page segmentation model. Regarding text classification features, we can clearly observe two different behaviours. On one hand, PGMs performed significantly better with RLF features than when regular Gabor features were used. Both *Body* and *Name* recognition classes always improved, where the improvement in *Name* F-measure is remarkable. The class *Tax* was the most challenging class. Although the differences in F-measure were small, we can see that Gabor features provided less precision but more recall whereas RLF produced higher precision and lower recall values.

On the other hand, the 2D-PCFG model obtained similar performance using both sets of features, and even results with Gabor features were slightly better than results provided by RLF features. 2D-PCFG is a powerful model that is able to take advantage of the knowledge about the document structure. Thus, grammars were able to overcome the lacks of Gabor features obtaining very good results for all classes without the additional spatial information provided by RLF. Given that we learnt probabilistic information about the structure of the documents from training data, the model was able to successfully parse documents using the regular Gabor features. Figure 8.14 shows an example of recognition using 2D-PCFG and both sets of features. We can see how the overlapping region between *Name* and *Body* is classified as *Body*. Also, as results pointed out (see Table 8.5), Gabor features obtained higher recall and we can see that resulting regions are larger. On the other hand, RLF provided higher precision by adjusting better the size of the regions detected. Finally, recognizing the space between records is difficult due to the large calligraphic letters considered as background.

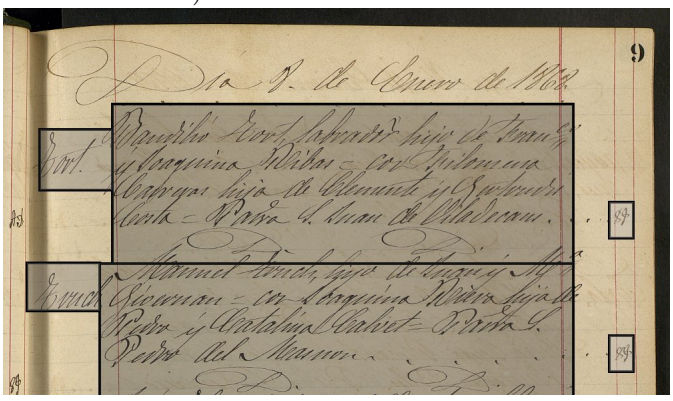
Comparing the performance of the different models, 2D-PCFG outperformed PGMs. Results showed that grammars achieved a great improvement even using Gabor features. The best results among the PGMs were obtained by ICM and RLF features with F-measure 0.91, 0.72 and 0.58 for classes *Body*, *Name* and *Tax*, respectively. The 2D-PCFG model with Gabor features achieved F-measure 0.93, 0.78 and 0.71 for classes *Body*, *Name* and *Tax*, respectively.

PGMs classify cells but they do not identify the explicit segmentation in records. However, the most likely hypothesis according to the 2D-PCFG model provides a derivation tree that accounts for both the structure of the document and the segmentation in cells. Using this information we extracted the number of records detected in each document. Then, we computed the percentage of documents in the test set where the number of records detected was correct.

a) Ground-truth



b) 2D-PCFG with Gabor



c) 2D-PCFG with RLF

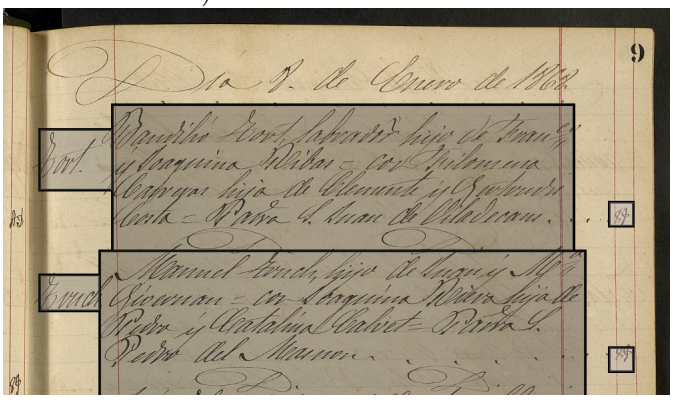


Figure 8.14: Example of page segmentation and structure detection with 2D-PCFG using cells of 50×50 pixels and different text classification features.

2D-PCFG in combination with Gabor features computed the right number of records in 80% of the test documents, whereas with RLF features only 52.5% of the documents had the correct number of records detected. All the errors were due to oversegmentation in both sets of features. This measure helps to assess the quality of the recognition such that we can see that in addition to the slight improvement of Gabor features with respect to RLF features, the number of records detected presented an important difference.

8.4.7 Summary

Given the grammar-based model developed in this thesis for math expression recognition, in this study we proposed an application of 2D-PCFG for page segmentation of structured documents. We evaluated this proposal using two sets of features: Gabor and RLF. We also tested several inference algorithms for PGMs, where RLF features obtained better results than Gabor features.

The experimentation carried out proved that 2D-PCFG parsing outperformed PGMs in this task. Furthermore, 2D-PCFG obtained better results with Gabor features than using RLF features, because the grammatical model was able to take advantage of the knowledge about the document structure. Moreover, grammars were able to provide the detailed and explicit information of the page segmentation, hence, record-level evaluation could be done and results also showed a good performance of the model.

Chapter

9 Conclusions

This thesis was devoted to develop a fully integrated approach for mathematical expression recognition. Along the chapters of this thesis, we presented an approach based on parsing 2D-PCFG, and we studied each of the specific tasks and issues related to automatic recognition of mathematical expressions. We evaluated the different proposals and methods described in this thesis with several experiments, and finally we explored some of the applications derived from the work developed during these years.

This final chapter summarizes the developments and conclusions of this thesis and we also discuss possible future research directions. Finally, we report the scientific contributions and publications that resulted from the work carried out in different tasks.

Chapter Outline

9.1 Summary

At the beginning of this thesis, in Chapter 1, we identified the problems of mathematical expression recognition and we stated generally the scientific goals we pursued along this document. The goals achieved can be grouped in three main tasks:

1. Develop an approach for automatic mathematical expression recognition.
2. Use public data and standard performance metrics.
3. Explore applications of the developed techniques.

Developing an approach for mathematical expression recognition was the main goal of this thesis. Although there are several options to tackle this problem, we also had specific objectives to develop our method. In this thesis, we presented an approach that meets the following features:

- We developed an integrated approach such that symbol segmentation, symbol recognition and structural analysis are simultaneously optimized.
- We defined a formal statistical framework based on parsing 2D-PCFG.
- The developed approach is generalized for any type of mathematical expression: printed or handwritten, online or offline.
- The system accept virtually any input expression, i.e. the expressions can be introduced in any order.
- All the probability distributions used in the statistical framework are estimated from data.

Recognizing mathematical expressions is a complex task that deals with several problems simultaneously. Furthermore, we generalized the formal framework for any type of expression and studied specific modules. Figure 9.1 presents an outline of the different components of our mathematical expression parser. We can see how all the parts contribute to the recognition process, as well as how the approach is generalized for any type of mathematical expression: online handwritten, offline handwritten, and offline printed.

In order to provide **comparable results**, we wanted to **use public data and standard performance metrics**. We believe it is of paramount importance to assess improvements in any field and advance towards the best solutions. Regarding this matter, we achieved the following results:

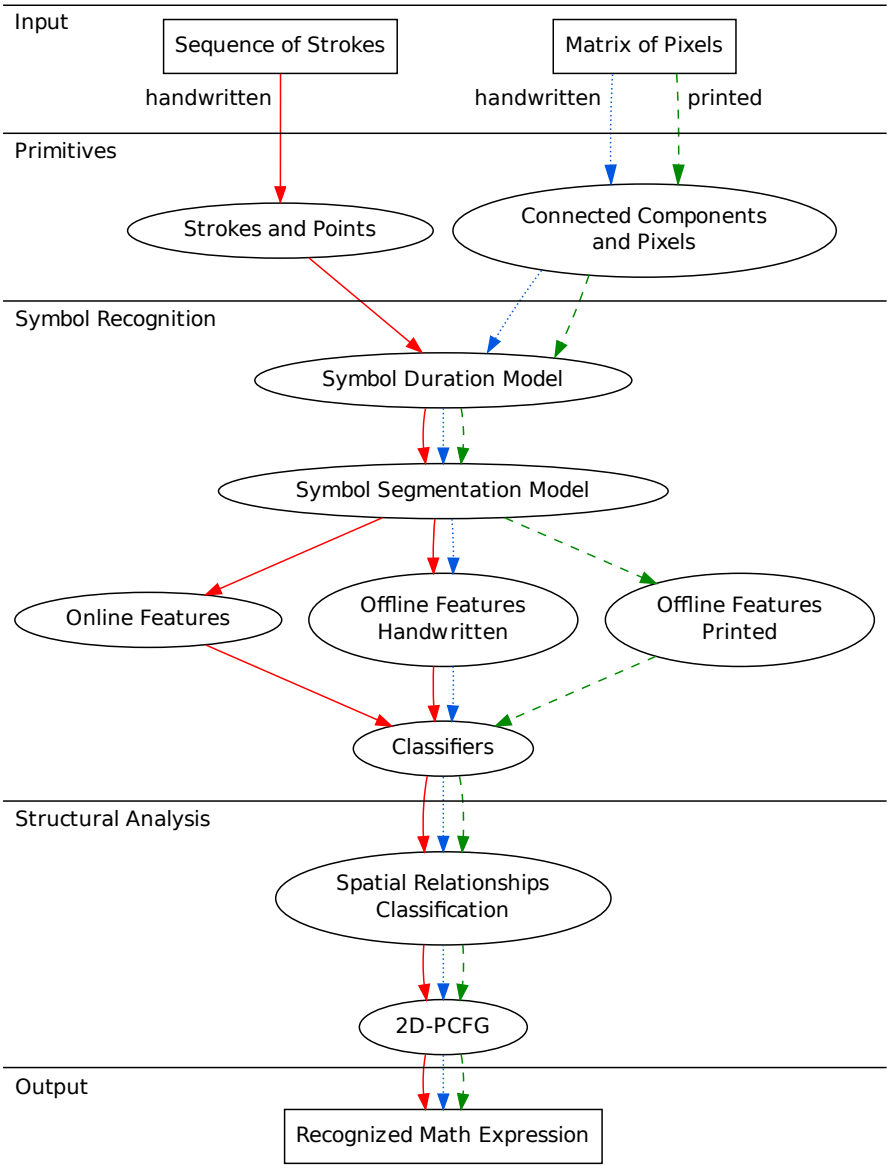


Figure 9.1: Diagram that summarizes the common and specific parts for recognizing any type of mathematical expression according to the approach developed in this thesis: online handwritten (solid), offline handwritten (dotted) and offline printed (dashed).

- We analyzed the problem of performance evaluation and proposed an automatic performance evaluation metric.
- We report results using public datasets and standard metrics, which allowed us to look for the best features, classifiers and methodologies.
- We released most of the software developed in this thesis as open-source.
- We participated in international competitions in order to evaluate our proposals and compare them to other solutions under the same conditions.

Once we have developed and evaluated our approach for mathematical expression recognition, we **explored several applications of the developed techniques**:

- The transcription of handwritten math expressions as a tool to introduce math notation into computer devices by means of handwriting. This transcription was used to perform simple computations or information retrieval using external services.
- μCAPTCHA, a novel method to tell humans and computers apart by means of math handwriting input. This application is also able to generate annotated online handwritten math expressions.
- A method for layout analysis based on the statistical framework for parsing 2D-PCFG developed in this thesis.

Finally, we identify the following **future work research directions** regarding mathematical expression recognition:

- Further research could be done on math symbol recognition, with new features and classifiers. In addition, two issues could be tackled. First, special treatment for recognition of similar shaped symbols ($\{x, \times, X\}$, $\{o, 0\}$, etc.). Second, tackle recognition of small symbols like punctuation marks.
- More advanced features and classifiers for symbol segmentation model. Our current model uses four simple features and GMM as classifiers, which are generative models. There are many features proposed for this task [Hu and Zanibbi, 2013] and discriminative training could provide better results.

- Spatial relationship classification has high error rates for distinguishing between right, superscript and subscript relations. It could be interesting to train different classifiers depending on symbol categories as it improves classification results between symbols [Aly et al., 2009].
- Further research should be done for the application of the integrated approach developed in this thesis to offline recognition, both printed and handwritten mathematical expressions.
- In many natural language processing problems, language models such as n-grams can be easily estimated from large resources, like collections of books, wikipedia, etc. An interesting future research direction is the estimation of language models for mathematical expression recognition using the resources available on Internet.
- We would like to explore the problem of math information retrieval, an interesting application of math expression recognition that has brought attention during the last years.

9.2 Scientific Contributions

The research work presented in this thesis has been developed during several years. We have tried to deal with most of the issues related to mathematical expression recognition and some of its applications. As a result, this thesis has generated the following contributions: 8 conference papers, 3 journal papers (plus one submitted), 3 competition awards and a technology transfer project. Below we sum up the scientific contributions grouped according to the different task within mathematical expression recognition.

Mathematical Symbol Classification

The study about different symbol classifiers and features for printed mathematical symbol classification was published in:

- Francisco Álvaro and Joan-Andreu Sánchez. Comparing Several Techniques for Offline Recognition of Printed Mathematical Symbols. *International Conference on Pattern Recognition (ICPR)*, 2010, pp. 1953–1956.

Regarding handwritten mathematical symbol recognition, the comparison between online features and hybrid features, as well as the performance evaluation of HMM and BLSTM-RNN was published in:

- Francisco Álvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. Classification of On-line Mathematical Symbols with Hybrid Features and Recurrent Neural Networks. *International Conference on Document Analysis and Recognition (ICDAR)*, 2013, pp. 1012–1016.

The comparison of different offline features for classifying handwritten math symbols using BLSTM-RNN and the combination of online and offline features was published in:

- Francisco Álvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. Offline Features for Classifying Handwritten Math Symbols with Recurrent Neural Networks. *International Conference on Pattern Recognition (ICPR)*, 2014, pp. 2944–2949.

Spatial Relationships Classification

A study about different features for spatial relationship classification was published in:

- Francisco Álvaro and Richard Zanibbi. A Shape-Based Layout Descriptor for Classifying Spatial Relationships in Handwritten Math. *ACM Symposium on Document Engineering (DocEng)*, 2013, pp. 123–126.

Automatic Performance Evaluation

We published two contributions regarding the problem of automatic performance evaluation. First, a study concerning the ambiguity in tree representation and unbiased evaluation by means of constrained parsing:

- Francisco Álvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. Unbiased Evaluation of Handwritten Mathematical Expression Recognition. *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2012, pp. 181–186.

Also, we proposed a performance evaluation metric based on image-matching and we presented it in:

- Francisco Álvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. An image-based measure for evaluation of mathematical expression recognition. *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, 2013, pp. 682–690.

Mathematical Expression Recognition

Our first research on mathematical expression recognition was focused on printed expressions. We published an initial model for parsing printed expressions based on parsing 2D-PCFG in :

- Francisco Álvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. Recognition of Printed Mathematical Expressions Using Two-dimensional Stochastic Context-Free Grammars. *International Conference on Document Analysis and Recognition (ICDAR)*, 2011, pp. 1225–1229.

Afterwards, we moved to handwritten mathematical expression recognition. Based on the work developed for printed expressions, we advanced and developed an approach for recognizing handwritten expressions. We published it in:

- 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*, 2014, vol. 35, pp. 58–67.

Finally, we developed the integrated approach presented in this thesis, and the description of this approach was submitted to:

- Francisco Álvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. An Integrated Grammar-based Approach for Mathematical Expression Recognition. *Pattern Recognition*, 2015, (submitted).

Applications

Some of the applications of mathematical expression recognition we explored in this thesis resulted in publications. The approach for layout analysis based on parsing 2D-PCFG of structured documents was initially published in:

- Francisco Álvaro, Francisco Cruz, Joan-Andreu Sánchez, Oriol Ramos Terrades and José-Miguel Benedí. Page Segmentation of Structured Documents Using 2D Stochastic Context-Free Grammars. *Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, 2013, pp. 133–140.

Later, we extended that research such that the application reported in this thesis was presented in:

- Francisco Álvaro, Francisco Cruz, Joan-Andreu Sánchez, Oriol Ramos Terrades and José-Miguel Benedí. Structure detection and segmentation of documents using 2D stochastic context-free grammars. *Neurocomputing*, 2015, vol. 150, pp. 147–154.

Finally, the μCAPTCHA method has been accepted for publication in the following journal:

- Luis A. Leiva and Francisco Álvaro. μCAPTCHA: Human Interaction Proofs Tailored to Touch-Capable Devices via Math Handwriting. *International Journal of Human-Computer Interaction*, 2015.

Awards

The Competition on Recognition of On-line Handwritten Mathematical Expression (CROHME) has been held since 2011 [Mouchère et al., 2011, 2012, 2013, 2014]. So far, we have participated in all editions of the competition and received the following awards:

- **CROHME 2011: Winner of the CROHME competition**. Mouchère H., Viard-Gaudin C., Garain U., Kim D. H., Kim J. H. CROHME 2011: Competition on Recognition of Online Handwritten Mathematical Expressions. *International Conference on Document Analysis and Recognition (ICDAR)* 2011. Beijing, China.

- **CROHME 2013: Best system using CROHME training set**. Mouchère H., Viard-Gaudin C., Zanibbi R., Garain U., Kim D. H., Kim J. H. ICDAR 2013 CROHME: Third International Competition on Recognition of Online Handwritten Mathematical Expressions. *International Conference on Document Analysis and Recognition (ICDAR)*, 2013. Washington, DC, USA.

- **CROHME 2014: Best system trained on CROHME dataset**. Mouchère H., Viard-Gaudin C., Zanibbi R., Garain U. ICFHR 2014 Competition on Recognition of On-line Handwritten Mathematical Expressions (CROHME 2014). *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014. Creta, Greece.

List of Acronyms

2D-PCFG	Two-Dimensional Probabilistic Context-Free Grammar
BIDM	Binary Image Distortion Model
BLSTM	Bidirectional Long Short-Term Memory
CAPTCHA	Completely Automated Public Turing Test to Tell Computers and Humans Apart
CFG	Context-Free Grammar
CNF	Chomsky Normal Form
CYK	Cocke-Younger-Kasami
DIA	Document Image Analysis
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
IDM	Image Distortion Model
IMEGE	Image-based Mathematical Expression Global Error
k-NN	k Nearest Neighbors
LSTM	Long Short-Term Memory
MLP	Multi Layer Perceptron
OCR	Optical Character Recognition
PCA	Principal Component Analysis
PCFG	Probabilistic Context-Free Grammar
PGM	Probabilistic Graphical Model
RNN	Recurrent Neural Network
SUS	System Usability Scale
SVM	Support Vector Machines
TLX	Task Load Index
WNN	Weighted Nearest Neighbor

List of Figures

List of Tables

List of Algorithms

Bibliography

F. Aguilar and N. S. T. Hirata. ExpressMatch: A System for Creating Ground-Truthed Datasets of Online Mathematical Expressions. In *Document Analysis Systems (DAS), 2012 10th IAPR International Workshop on*, pages 155–159, March 2012.

F. Álvaro, J. Sánchez, and J. Benedí. Recognition of printed mathematical expressions using two-dimensional stochastic context-free grammars. In *International Conference on Document Analysis and Recognition*, pages 1225–1229, 2011.

F. Álvaro, J.-A. Sánchez, and J.-M. Benedí. Unbiased evaluation of handwritten mathematical expression recognition. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 181–186, 2012.

F. Álvaro, J.-A. Sánchez, and J.-M. Benedí. Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and hidden markov models. *Pattern Recognition Letters*, 35(0):58–67, 2014.

W. Aly, S. Uchida, and M. Suzuki. Automatic classification of spatial relationships among mathematical symbols using geometric features. *IEICE Transactions*, 92-D(11):2235–2243, 2009.

R. H. Anderson. Syntax-directed recognition of hand-printed two-dimensional mathematics. In *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the ACM Symposium*, pages 436–459. ACM, 1967.

A. Antonacopoulos, C. Clausner, C. Papadopoulos, and S. Pletschacher. Historical document layout analysis competition. In *Proc. of ICDAR*, pages 1516–1520, Beijing, China, 2011.

A. Antonacopoulos, C. Clausner, C. Papadopoulos, and S. Pletschacher. ICDAR 2013 Competition on Historical Newspaper Layout Analysis (HNLA

2013). In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 1454–1458, Aug 2013a.

A. Antonacopoulos, C. Clausner, C. Papadopoulos, and S. Pletschacher. ICDAR 2013 Competition on Historical Book Recognition (HBR 2013). In *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*, pages 1459–1463, Aug 2013b.

A.-M. Awal, H. Mouchere, and C. Viard-Gaudin. Towards handwritten mathematical expression recognition. In *International Conference on Document Analysis and Recognition*, pages 1046–1050, 2009.

A.-M. Awal, H. Mouchere, and C. Viard-Gaudin. The problem of handwritten mathematical expression recognition evaluation. *Frontiers in Handwriting Recognition, International Conference on*, 0:646–651, 2010.

A.-M. Awal, H. Mouchère, and C. Viard-Gaudin. A global learning approach for an online handwritten mathematical expression recognition system. *Pattern Recognition Letters*, 35(0):68 – 77, 2014. Frontiers in Handwriting Processing.

H. S. Baird and J. L. Bentley. Implicit CAPTCHAs. In *Proc. SPIE/IST*, pages 507–518, 2005.

H. S. Baird and K. Popat. Human interactive proofs and document image analysis. In *Proc. DAS*, pages 507–518, 2002.

A. Basso and F. Bergadano. Anti-bot strategies based on human interactive proofs. In *Handbook of Information and Communication Security*, pages 273–291. 2010.

L. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process. *Inequalities*, pages 1–8, 1972.

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):509–522, 2002.

J. Besag. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society. Series B (Methodological)*, 48(3):259–302, 1986.

C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

J. Blocki, M. Blum, and A. Datta. GOTCHA password hackers! In *Proceedings of the ACM Workshop on Artificial Intelligence and Security (AISec)*, pages 25–34, 2013.

Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 23 (11):1222–1239, 2001.

M. Bulacu, R. Koert, L. Schomaker, and T. Zant. Layout analysis of handwritten historical documents for searching the archive of the cabinet of the Dutch queen. In *Proc. of ICDAR*, volume 1, pages 23–26, Brazil, 2007.

E. Bursztein, A. Moscicki, C. Fabry, S. Bethard, J. C. Mitchell, and D. Jurafsky. Easy does it: More usable CAPTCHAs. In *Proceedings of the SIGCHI conference on Human Factors in Computing systems (CHI)*, pages 2637–2646, 2014.

K.-F. Chan and D.-Y. Yeung. Mathematical expression recognition: a survey. *International Journal on Document Analysis and Recognition*, 3:3–15, 2000.

K.-F. Chan and D.-Y. Yeung. Error detection, error correction and performance evaluation in on-line mathematical expression recognition. *Pattern Recognition*, 34:1671 – 1684, 2001.

K. Chellapilla and P. Simard. Using machine learning to break visual human interaction proofs (HIPs). In *Proceedings of Neural Information Processing Systems (NIPS)*, 2005.

K. Chellapilla, K. Larson, P. Simard, and M. Czerwinski. Designing human friendly human interaction proofs (HIPs). In *Proceedings of the SIGCHI conference on Human Factors in Computing systems (CHI)*, pages 711–720, 2005.

T. Chen, Y. Yesilada, and S. Harper. What input errors do you experience? typing and pointing errors of mobile web users. *Int. J. Hum.-Comput. Stud.*, 68(3):138–157, 2010.

M. Chew and H. S. Baird. BaffleText: a human interactive proof. In *Proc. SPIE/IST*, 2003.

P. A. Chou. Recognition of equations using a two-dimensional stochastic context-free grammar. In W. A. Pearlman, editor, *Visual Communications and Image Processing IV*, volume 1199 of *SPIE Proceedings Series*, pages 852–863, 1989.

R. Chow, P. Golle, M. Jakobsson, L. Wang, and X. Wang. Making CAPTCHAs clickable. In *Proc. HotMobile*, pages 91–94, 2008.

A. L. Coates, H. Baird, and R. Faternan. Pessimal print: A reverse turing test. In *Proceedings of the Intl. Conf. on Document Analysis and Recgnition (ICDAR)*, pages 1154–1158, 2001.

G. F. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artif. Intell.*, 42(2-3):393–405, 1990.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.*, 2:265–292, 2002.

S. Crespi Reghizzi and M. Pradella. A CKY parser for picture grammars. *Information Processing Letters*, 105(6):213–217, Feb. 2008.

F. Cruz and O. R. Terrades. Document segmentation using relative location features. In *Proc. of ICPR*, pages 1562–1565, Japan, 2012.

K. Davila, S. Ludi, and R. Zanibbi. Using Off-line Features and Synthetic Data for On-line Handwritten Math Symbol Recognition. In *Frontiers in Handwriting Recognition (ICFHR), International Conference on*, pages 323–328, 2014.

P. Doetsch, M. Hamdani, H. Ney, A. Giménez, J. Andrés-Ferrer, and A. Juan. Comparison of Bernoulli and Gaussian HMMs Using a Vertical Repositioning Technique for Off-Line Handwriting Recognition. In *Frontiers in Handwriting Recognition (ICFHR), International Conference on*, pages 3–7, 2012.

P. Dreuw, D. Rybach, C. Gollan, and H. Ney. Writer adaptive training and writing variant model refinement for offline arabic handwriting recognition. In *Document Analysis and Recognition. International Conference on*, pages 21–25, 2009.

J. Elson, J. R. Douceur, J. Howell, and J. Saul. ASIRRA: A CAPTCHA that exploits interest-aligned manual image categorization. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 366–374, 2009.

A. Esteve, C. Cortina, and A. Cabré. Long term trends in marital age homogamy patterns: Spain, 1992-2006. *Population*, 64(1):173–202, 2009.

Y. Eto and M. Suzuki. Mathematical formula recognition using virtual link network. In *International Conference on Document Analysis and Recognition*, pages 762–767, Washington, DC, 2001.

R. Fateman, T. Tokuyasu, B. P. Berman, and N. Mitchell. Optical character recognition and parsing of typeset mathematics. *Journal of Visual Communication and Image Representation*, 7:2–15, 1996.

C. Faure and Z. Wang. Automatic perception of the structure of handwritten mathematical expressions. In *Computer Processing of Handwriting*, pages 337–361, 1990.

I. Fogel and D. Sagi. Gabor filters as texture discriminator. *Biological Cybernetics*, 61(2):103–113, 1989.

U. Garain and B. Chaudhuri. Recognition of online handwritten mathematical expressions. *IEEE Trans. on Systems, Man, and Cybernetics - Part B: Cybernetics*, 34(6):2366–2376, 2004.

U. Garain and B. Chaudhuri. A corpus for ocr research on mathematical expressions. *International Journal on Document Analysis and Recognition*, 7:241–259, 2005a.

U. Garain and B. Chaudhuri. Segmentation of touching symbols for ocr of printed mathematical expressions: an approach based on multifactorial analysis. In *International Conference on Document Analysis and Recognition*, pages 177–181 Vol. 1, Aug 2005b.

U. Garain, B. Chaudhuri, and R. Ghosh. A multiple-classifier system for recognition of printed mathematical symbols. In *Proc. ICPR*, volume 1, pages 380–383, 2004.

A. Giménez, I. Khoury, and A. Juan. Windowed Bernoulli Mixture HMMs for Arabic Handwritten Word Recognition. In *Frontiers in Handwriting Recognition (ICFHR), International Conference on*, pages 533–538, 2010.

I. J. Goodfellow, Y. Bulatov, J. Ibarz, S. Arnoud, and V. Shet. Multi-digit number recognition from street view imagery using deep convolutional neural networks. 2014.

J. Goodman. Semiring parsing. *Computational Linguistics*, 25(4):573–605, 1999.

A. Goshtasby. Description and discrimination of planar shapes using shape matrices. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-7(6):738–743, 1985.

R. Gossweiler, M. Kamvar, and S. Baluja. What's up CAPTCHA?: A CAPTCHA based on image orientation. In *Proceedings of the WWW*, pages 841–850, 2009.

S. Gould, J. Rodgers, D. Cohen, G. Elidan, and D. Koller. Multi-class segmentation with relative location prior. *Int. Journal of Computer Vision*, 80 (3):300–316, 2008.

A. Graves. RNNLIB: A recurrent neural network library for sequence learning problems. http://sourceforge.net/projects/rnnl/, 2010.

A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31 (5):855–868, 2009.

J. Ha, R. Haralick, and I. Phillips. Understanding mathematical expressions from document images. In *International Conference on Document Analysis and Recognition*, volume 2, pages 956–959, Aug 1995.

J. Handley, A. Namboodiri, and R. Zanibbi. Document understanding system using stochastic context-free grammars. *Proc. of ICDAR*, 1:511–515, 2005.

C. J. Hernandez-Castro and A. Ribagorda. Pitfalls in CAPTCHA design and implementation: The Math CAPTCHA, a case study. *Computers & Security*, 29(1):141–157, 2010.

J. M. G. Hidalgo and G. Alvarez. CAPTCHAs: An artificial intelligence application to web security. *Advances in Computers*, 83(1):109–181, 2011.

N. S. T. Hirata and W. Y. Honda. Automatic labeling of handwritten mathematical symbols via expression matching. In *International Conference on Graph-based Representations in Pattern Recognition*, pages 295–304, Munich, Germany, 2011.

J. P. Hourcade and T. R. Berkel. Simple pen interaction performance of young and older adults using handheld computers. *Interacting with Computers*, 20 (1):166–183, 2008.

L. Hu and R. Zanibbi. HMM-Based Recognition of Online Handwritten Mathematical Symbols Using Segmental K-Means Initialization and a Modified Pen-Up/Down Feature. *International Conference on Document Analysis and Recognition*, 0:457–462, 2011.

L. Hu and R. Zanibbi. Segmenting Handwritten Math Symbols Using AdaBoost and Multi-Scale Shape Context Features. In *International Conference on Document Analysis and Recognition*, 2013.

L. Hu, K. Hart, R. Pospesel, and R. Zanibbi. Baseline extraction-driven parsing of handwritten mathematical expressions. In *International Conference on Pattern Recognition*, pages 326–330, Nov 2012.

J. Ilonen, J.-K. Kamarainen, and H. Kälviäinen. Fast extraction of multi-resolution Gabor features. In *14th International Conference on Image Analysis and Processing*, pages 481–486, Modena, Italy, 2007.

A. Jain, A. Namboodiri, and J. Subrahmonia. Structure in online documents. In *Proc. of ICDAR*, volume 1, pages 844–848, 2001.

N. Jiang and F. Tian. A novel gesture-based CAPTCHA design for smart devices. In *Proc. BCS HCI*, page 49, 2013.

M. Jordan, Z. Ghahramani, T. Jaakkola, and L. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, Nov. 1999.

F. Julca-Aguilar, N. S. Hirata, C. Viard-Gaudin, H. Mouchère, and S. Medjkoune. Mathematical symbol hypothesis recognition with rejection option. In *Frontiers in Handwriting Recognition (ICFHR), International Conference on*, pages 500–504, 2014.

D. Jurafsky and J. H. Martin. *Speech and Language Processing (2nd edition) (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 2008.

B. Keshari and S. Watt. Hybrid mathematical symbol recognition using support vector machines. In *International Conference on Document Analysis and Recognition*, volume 2, pages 859 –863, 2007.

D. Keysers, T. Deselaers, C. Gollan, and H. Ney. Deformation models for image recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(8):1422–1435, 2007.

W. Kienzle and K. Hinckley. Writing handwritten messages on a small touchscreen. In *International Conference on Human-Computer Interaction with Mobile Devices and Services (mobileHCI)*, 2013.

K. A. Kluever and R. Zanibbi. Balancing usability and security in a video CAPTCHA. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS)*, pages 14:1–14:11, 2009.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

A. Kosmala and G. Rigoll. On-line handwritten formula recognition using statistical methods. In *International Conference on Pattern Recognition*, pages 1306–1308, 1998.

S. Kumar, R. Gupta, N. Khanna, S. Chaudhury, and S. D. Joshi. Text extraction and document image segmentation using matched wavelets and mrf model. *Image Processing, IEEE Transactions on Image Processing*, 16 (8):2117–2128, aug. 2007.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, pages 282–289, USA, 2001.

A. Lapointe and D. Blostein. Issues in performance evaluation: A case study of math recognition. In *International Conference on Document Analysis and Recognition*, pages 1355–1359, 2009.

S. Lavirotte and L. Pottier. Mathematical formula recognition using graph grammar. In *Proceedings of the SPIE*, volume 3305, pages 44–52, 1998.

H.-J. Lee and M.-C. Lee. Understanding mathematical expressions in a printed document. In *Proceedings of the Second International Conference on Document Analysis and Recognition*, pages 502–505, Oct 1993.

S. Lehmberg, H.-J. Winkler, and M. Lang. A soft-decision approach for symbol segmentation within handwritten mathematical expressions. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3434–3437 vol. 6, May 1996.

S. Z. Li. *Markov Random Field Modeling in Computer Vision*. Springer-Verlag, London, UK, UK, 1995.

L. Likforman-Sulem, A. Zahour, and B. Taconet. Text line segmentation of historical documents: a survey. *International Journal of Document Analysis and Recognition*, 9:123–138, 2007.

R. Lin, S.-Y. Huang, G. B. Bell, and Y.-K. Lee. A new CAPTCHA interface design for mobile devices. In *Proc. AUIC*, 2011.

M. Liwicki and H. Bunke. Feature selection for hmm and blstm based handwriting recognition of whiteborad notes. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(05):907–923, 2009.

Z. Luo, Y. Shi, and F. Soong. Symbol graph based discriminative training and rescoring for improved math symbol recognition. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1953–1956, 2008.

S. MacLean and G. Labahn. Elastic matching in linear time and constant space. *Document Analysis Systems, Ninth IAPR Workshop on (short paper)*, 2010.

S. MacLean and G. Labahn. A new approach for recognizing handwritten mathematics using relational grammars and fuzzy sets. *International Journal on Document Analysis and Recognition*, 16(2):139–163, 2013.

S. MacLean, G. Labahn, E. Lank, M. Marzouk, and D. Tausky. Grammar-based techniques for creating ground-truthed sketch corpora. *International Journal on Document Analysis and Recognition*, 14:65–74, 2011.

C. Malon, S. Uchida, and M. Suzuki. Mathematical symbol recognition with support vector machines. *Pattern Recognition Letters*, 29(9):1326–1332, 2008.

C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

S. Marinai, B. Miotti, and G. Soda. Using earth mover's distance in the bag-of-visual-words model for mathematical symbol retrieval. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1309–1313, 2011.

U.-V. Marti and H. Bunke. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *International Journal of Pattern Recognition and Artificial Intelligence*, 15 (01):65–90, 2001.

N. Matsakis. Recognition of handwritten mathematical expressions. Master's thesis, Massachusetts Institute of Technology, Cambridge, USA, May 1999.

M. Mohamed, N. Sachdeva, M. Georgescu, S. Gao, N. Saxena, C. Zhang, P. Kumaraguru, P. C. van Oorschot, and W.-B. Chen. A three-way investigation of a game-CAPTCHA: Automated attacks, relay attacks and usability. In *Proceedings of the Symposium on Information, Computer and Communications Security (CCS)*, pages 195–206, 2014.

H. Mouchère, C. Viard-Gaudin, U. Garain, D. Kim, and J. Kim. CROHME2011: Competition on Recognition of Online Handwritten Mathematical Expressions. In *Proceedings of the 11th International Conference on Document Analysis and Recognition, ICDAR 2011*, September 2011.

H. Mouchère, C. Viard-Gaudin, D. Kim, J. Kim, and U. Garain. ICFHR 2012 Competition on Recognition of On-Line Mathematical Expressions (CROHME 2012). In *Frontiers in Handwriting Recognition (ICFHR), International Conference on*, pages 811–816, Sept 2012.

H. Mouchère, C. Viard-Gaudin, R. Zanibbi, U. Garain, and D. H. Kim. ICDAR 2013 CROHME: Third International Competition on Recognition of Online Handwritten Mathematical Expressions. In *Document Analysis and Recognition, International Conference on*, pages 1428–1432, 2013.

H. Mouchère, C. Viard-Gaudin, R. Zanibbi, and U. Garain. ICFHR 2014 Competition on Recognition of On-Line Handwritten Mathematical Expressions (CROHME 2014). In *Frontiers in Handwriting Recognition (ICFHR), International Conference on*, pages 791–796, Sept 2014.

J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

H. Ney. Stochastic Grammars and Pattern Recognition. In P. Laface and R. De Mori, editors, *Speech Recognition and Understanding*, volume 75, pages 319–344. Springer Berlin Heidelberg, 1992.

A. Nomura, K. Michishita, S. Uchida, and M. Suzuki. Detection and segmentation of touching characters in mathematical expressions. In *International Conference on Document Analysis and Recognition*, pages 126–130 vol.1, Aug 2003.

M. Okamoto and A. Miyazawa. An experimental implementation of a document recognition system for papers containing mathematical expressions. In *Structured Document Image Analysis*, pages 36–53. 1992.

M. Okamoto, H. Imai, and K. Takagi. Performance evaluation of a robust method for mathematical expression recognition. In *Proc. 6th International Conference on Document Analysis and Recognition (ICDAR'01)*, pages 121–128, September 2001.

N. Okamoto and M. B. Recognition of mathematical expressions by using the layout structures of symbols. In *International Conference on Document Analysis and Recognition*, pages 242–250, 1991.

N. Otsu. A Threshold Selection Method from Gray-level Histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, 1979.

L. Ouyang and R. Zanibbi. Identifying layout classes for mathematical symbols using layout context. In *IEEE Western New York Image Processing Workshop*, 2009.

R. Paredes and E. Vidal. Learning weighted metrics to minimize nearest-neighbor classification error. *IEEE Transaction on Pattern Analisys and Machine Intelligence*, 28(7), 2006.

P. Pavan Kumar, A. Agarwal, and C. Bhagvati. A string matching based algorithm for performance evaluation of mathematical expression recognition. *Sadhana*, 39(1):63–79, 2014.

B. Pearlmutter. Learning state space trajectories in recurrent neural networks. In *International Joint Conference on Neural Networks*, volume 2, pages 365–372, 1989.

I. Phillips. Methodologies for using UW databases for OCR and image understanding systems. In *Proc. SPIE, Document Recognition V*, volume 3305, pages 112–127, 1998.

R. Plamondon and S. Srihari. Online and off-line handwriting recognition: a comprehensive survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(1):63–84, 2000.

D. Průša and V. Hlaváč. Mathematical formulae recognition using 2d grammars. *International Conference on Document Analysis and Recognition*, 2: 849–853, 2007.

J. Puigcerver, A. H. Toselli, and E. Vidal. Word-graph and character-lattice combination for kws in handwritten documents. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 181–186, 2014.

S. Quiniou, H. Mouchere, S. Saldarriaga, C. Viard-Gaudin, E. Morin, S. Petitrenaud, and S. Medjkoune. HAMEX - A Handwritten and Audio Dataset of Mathematical Expressions. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 452–456, Sept 2011.

V. Romero, A. Fornés, N. Serrano, J. Sánchez, A. Toselli, V. Frinken, E. Vidal, and J. Lladós. The ESPOSALLES database: An ancient marriage license corpus for off-line handwriting recognition. *Pattern Recognition*, 46: 1658–1669, 2013.

N. Roshanbin and J. Miller. A survey and analysis of current CAPTCHA approaches. *J. Web Eng.*, 12(1-2):1–40, 2013.

S. A. Ross, J. A. Halderman, and A. Finkelstein. Sketcha: A CAPTCHA based on line drawings of 3D models. pages 821–830, 2010.

D. Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1-2):273–302, 1996.

A. Rusu and V. Govindaraju. Handwritten CAPTCHA: Using the difference in the abilities of humans and machines in reading handwritten words. In *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pages 226–231, 2004.

A. Rusu, R. Docimo, and A. Rusu. Leveraging cognitive factors in securing WWW with CAPTCHA. In *Proceedings of the USENIX Conference on Web Application Development (WebApps)*, pages 5–5, 2010.

K. Sain, A. Dasgupta, and U. Garain. EMERS: a tree matching based performance evaluation of mathematical expression recognition systems. *IJDAR*, pages 1–11, 2010.

M. Schuster and K. Paliwal. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*, 45(11):2673–2681, 1997.

F. Shafait, D. Keysers, and T. Breuel. Performance evaluation and benchmarking of six-page segmentation algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):941–954, 2008.

Y. Shi and F. Soong. A symbol graph based handwritten math expression recognition. In *International Conference on Pattern Recognition*, pages 1–4, 2008.

Y. Shi, H. Li, and F. K. Soong. A Unified Framework for Symbol Segmentation and Recognition of Handwritten Mathematical Expressions. In *International Conference on Document Analysis and Recognition*, pages 854–858, Washington, DC, 2007.

M. Shirali-Shahreza and S. Shirali-Shahreza. Drawing CAPTCHA. In *Proc. ITI*, pages 475–480, 2006.

M. Shirali-Shahreza and S. Shirali-Shahreza. Motion CAPTCHA. In *Proc. HSI*, pages 1042–1044, 2008.

S. Shirali-Shahreza and M. Shirali-Shahreza. Multilingual highlighting CAPTCHA. In *Proc. ITNG*, pages 447–452, 2011.

S. Shirali-Shahreza, G. Penn, R. Balakrishnan, and Y. Ganjali. SeeSay and HearSay CAPTCHA for mobile interaction. In *Proceedings of the SIGCHI conference on Human Factors in Computing systems (CHI)*, pages 2147–2156, 2013.

R. Sibson. SLINK: an optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34, 1973.

P. Y. Simard, R. Szeliski, J. Benaloh, J. Couvreur, and I. Calinov. Using character recognition and segmentation to tell computer from humans. In *Proceedings of the Intl. Conf. on Document Analysis and Recgnition (ICDAR)*, page 418, 2003.

F. Simistira, V. Katsouros, and G. Carayannis. A Template Matching Distance for Recognition of On-Line Mathematical Symbols. In *Frontiers in Handwriting Recognition (ICFHR), International Conference on*, pages 415–420, 2008.

F. Simistira, V. Papavassiliou, V. Katsouros, and G. Carayannis. Recognition of spatial relations in mathematical formulas. In *Frontiers in Handwriting Recognition (ICFHR), International Conference on*, pages 164–168, 2014.

D. Stalnaker and R. Zanibbi. Math expression retrieval using an inverted index over symbol pairs. In *Document Recognition and Retrieval (DRR)*, 2015.

J. Stria, M. Bresler, D. Prusa, and V. Hlavac. MfrDB: Database of Annotated On-Line Mathematical Formulae. In *Proceedings of the 2012 International Conference on Frontiers in Handwriting Recognition*, ICFHR '12, pages 542–547, Washington, DC, USA, 2012. IEEE Computer Society.

B. Su, X. Ding, L. Peng, and C. Liu. A novel baseline-independent feature set for arabic handwriting recognition. In *Document Analysis and Recognition, International Conference on*, pages 1250–1254, 2013.

M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, and T. Kanahori. Infty- an integrated ocr system for mathematical documents. In C. Vanoirbeek, C. Roisin, and E.Munson, editors, *Proc. of ACM Symposium on Document Engineering*, pages 95–104, Grenoble, 2003.

M. Suzuki, S. Uchida, and A. Nomura. A ground-truthed mathematical character and symbol image database. In *International Conference on Document Analysis and Recognition*, pages 675–679 Vol. 2, Aug 2005.

E. Tapia and R. Rojas. Recognition of on-line handwritten mathematical expressions using a minimum spanning tree construction and symbol dominance. In *Graphics Recognition. Recent Advances and Perspectives*, volume 3088 of *Lecture Notes in Computer Science*, pages 329–340. Springer Berlin Heidelberg, 2004.

A. Thammano and S. Rugkunchon. A neural network model for online handwritten mathematical symbol recognition. In D.-S. Huang, K. Li, and G. Irwin, editors, *Intelligent Computing*, volume 4113 of *Lecture Notes in Computer Science*, pages 292–298. Springer Berlin / Heidelberg, 2006.

X. Tian and Y. Zhang. Segmentation of touching characters in mathematical expressions using contour feature technique. In *ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, volume 1, pages 206–209, July 2007.

A. Toselli, A. Juan, and E. Vidal. Spontaneous handwriting recognition and classification. In *Proc. of the 17th International Conference on Pattern Recognition*, pages 433–436, Cambridge, UK, August 2004a.

A. Toselli, M. Pastor, and E. Vidal. On-line handwriting recognition system for tamil handwritten characters. In *Pattern Recognition and Image Analysis*, volume 4477 of *Lecture Notes in Computer Science*, pages 370–377. Springer Berlin / Heidelberg, 2007.

A. H. Toselli, A. Juan, D. Keysers, J. González, I. Salvador, H. Ney, E. Vidal, and F. Casacuberta. Integrated Handwriting Recognition and Interpretation using Finite-State Models. *IJPRAI*, 18(4):519–539, June 2004b.

K. Toyozumi, N. Yamada, T. Kitasaka, K. Mori, Y. Suenaga, K. Mase, and T. Takahashi. A study of symbol segmentation method for handwritten mathematical formula recognition using mathematical structure information. In *International Conference on Pattern Recognition*, volume 2, pages 630–633, Aug 2004.

H. Twaakyondo and M. Okamoto. Structure analysis and recognition of mathematical expressions. In *International Conference on Document Analysis and Recognition*, volume 1, pages 430–437, Aug 1995.

L. von Ahn, M. Blum, and J. Langford. Telling humans and computers apart automatically. *Communications of the ACM*, 47(2):56–60, 2004.

L. von Ahn, B. Maurer, C. McMillen, D. Abraham, and M. Blum. re-CAPTCHA: Human-based character recognition via web security measures. *Science*, 321(5895), 2008.

M. J. Wainwright and M. I. Jordan. *Graphical Models, Exponential Families, and Variational Inference*, volume 1. Now Publishers Inc., Hanover, MA, USA, Jan. 2008.

Y. Weiss. Correctness of local probability propagation in graphical models with loops, 2000.

F. Wessel, R. Schluter, K. Macherey, and H. Ney. Confidence measures for large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, 9(3):288–298, Mar 2001.

H.-J. Winkler. HMM-based handwritten symbol recognition using on-line and off-line features. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 6, pages 3438–3441, 1996.

J. O. Wobbrock, A. D. Wilson, and Y. Li. Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In *Proceedings of Annual ACM Symposium on User Interface Software and Technology (UIST)*, pages 159–168, 2007.

Y. Xu, G. Reynaga, S. Chiasson, J.-M. Frahm, F. Monrose, and P. Van Oorschot. Security and usability challenges of moving-object CAPTCHAs: Decoding codewords in motion. In *Proceedings of the USENIX Conference on Security Symposium*, pages 4–20, 2012.

R. Yamamoto, S. Sako, T. Nishimoto, and S. Sagayama. On-line recognition of handwritten mathematical expressions based on stroke-based stochastic context-free grammar. *IEIC Technical Report*, 2006.

J. Yan and A. S. El Ahmad. Usability of CAPTCHAs or usability issues in CAPTCHA design. In *Proceedings of the Symposium on Usable Privacy and Security (SOUPS)*, pages 44–52, 2008.

M. Yang, K. Kpalma, and J. Ronsin. A Survey of Shape Feature Extraction Techniques. In P.-Y. Yin, editor, *Pattern Recognition*, pages 43–90. Nov. 2008.

J. S. Yedidia, W. T. Freeman, and Y. Weiss. Exploring artificial intelligence in the new millennium. chapter Understanding Belief Propagation and Its Generalizations, pages 239–269. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

R. Zanibbi and D. Blostein. Recognition and retrieval of mathematical expressions. *International Journal on Document Analysis and Recognition*, 15 (4):331–357, 2012.

R. Zanibbi, D. Blostein, and J. Cordy. Recognizing mathematical expressions using tree transformation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 24(11):1–13, 2002.

R. Zanibbi, A. Pillay, H. Mouchère, C. Viard-Gaudin, and D. Blostein. Stroke-based performance metrics for handwritten mathematical expressions. In *International Conference on Document Analysis and Recognition*, pages 334–338, Sept 2011.

R. Zanibbi, H. Mouchère, and C. Viard-Gaudin. Evaluating structural pattern recognition for handwritten math via primitive label graphs. In *Document Recognition and Retrieval (DRR)*, 2013.

M. Zelkowitz. *Security on the Web*. Academic Press, 2001.

L. Zhang, D. Blostein, and R. Zanibbi. Using fuzzy logic to analyze superscript and subscript relations in handwritten mathematical expressions. In *International Conference on Document Analysis and Recognition*, pages 972–976 Vol. 2, Aug 2005.