Applying Hierarchical Contextual Parsing with Visual Density and Geometric Features to Typeset Formula Recognition

APPROVED BY

SUPERVISING COMMITTEE:

Dr. Richard Zanibbi, Supervisor

Zack Butler, Reader

Ivona Bezakova, Observer

Applying Hierarchical Contextual Parsing with Visual Density and Geometric Features to Typeset Formula Recognition

by

Michael Patrick Erickson Condon, B.S.

THESIS

Presented to the Faculty of the Computer Science Department

Golisano College of Computer and Information Sciences

Rochester Institute of Technology

in Partial Fulfillment

of the Requirements

for the Degree of

Master of Science

Rochester Institute of Technology

August 2017

Acknowledgments

First I would like to give the greatest thanks to my advisor Dr. Richard Zanibbi for all his help in my academic pursuits and beyond. This includes but is not limited to welcoming me into his lab, supporting me in my work, and unending amounts of guidance.

Also I would like to thank my thesis committee Dr. Ivona Bezàkovà and Dr. Zack Butler for both serving on my thesis committee and their memorable contributions to my studies at Rochester Institute of Technology.

I would also like to thank Dr. Suzuki and the Infty research team for making their datasets available to me, answering my questions, and allowing for the free release of the modified dataset.

I wish to also thank all the members of the Document and Pattern Recognition Lab at Rochester Institute of Technology. First Chinmay Jain and Lakshmi Ravi with whom I built the $Pythagor^m$ system and without whom I could not have completed my research. Lakshmi deserves additional thanks for the shared ideas and advice, in particular using shape of symbols in relationship classification. Next is Kenny Davila for his substantial contributions to the $Pythagor^m$ system and added guidance during my time in the lab.

For others at Rochester Institute of Technology I would like to thank my fellow teaching assistants and our ring leader Prof. Sean Strout for their support and friendship.

Finally I would like to thank my parents, my brother, and the rest of my family for all the years of love and support.

Abstract

Applying Hierarchical Contextual Parsing with Visual Density and Geometric Features to Typeset Formula Recognition

Michael Patrick Erickson Condon, M.S.
Rochester Institute of Technology, 2017

Supervisor: Dr. Richard Zanibbi

We demonstrate that recognition of scanned typeset mathematical expression images can be done by extracting maximum spanning trees from line of sight graphs weighted using geometric and visual density features. The approach used is hierarchical contextual parsing (HCP): Hierarchical in terms of starting with connected components and building to the symbol level using visual, spatial, and contextual features of connected components. Once connected components have been segmented into symbols, a new set of spatial, visual, and contextual features are extracted. One set of visual features is used for symbol classification, and another for parsing. The features are used in parsing to assign classifications and confidences to edges in a line of sight symbol graph. Layout trees describe expression structure in terms of spatial relations between symbols, such as horizontal, subscript, and superscript. From the weighted graph Edmonds' algorithm is used to extract a maximum spanning tree. Segmentation and parsing are done without using symbol classification information, and symbol classification is done independently of expression structure recognition. The commonality between the recognition processes is the type of features they use, the visual densities. These visual densities are used for shape, spatial, and contextual information. The contextual information is shown to help in segmentation, parsing, and symbol recognition.

The hierarchical contextual parsing has been implemented in the Python and Graph-based Online/Offline Recognizer for Math ($Pythagor^m$) system and tested on the InftyMCCDB-2 dataset. We created InftyMCCDB-2 from InftyCDB-2 as a open source dataset for scanned typeset math expression recognition. In building InftyMCCDB-2 modified formula structure representations were used to better capture the spatial positioning of symbols in the expression structures. Namely, baseline punctuation and symbol accents were moved out of horizontal baselines as their positions are not horizontally aligned with symbols on a writing line. With the transformed spatial layouts and HCP, 95.97% of expressions were parsed correctly when given symbols and 93.95% correctly parsed when requiring symbol segmentation from connected components. Overall HCP reached 90.83% expression recognition rate from connected components.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

There is a growing interest in the problem of math expression recognition. Math recognition is a language problem with a high level of difficulty. Standard written language has a very regular structure, linear lines arranged in parallel. In addition written language has a more restricted set of symbols. Math expressions possess complex two dimensional symbol arrangements with recursive or hierarchical structures. Examination of the mathematical expression problem naturally leads to approaches with application for similar complex or simple language problems, like written language and structure formulae in chemistry.

This problem looks at detecting symbols, classifying individual symbols, and determining the structure of the expression containing the symbols. detecting symbols includes segmenting the primitive structures, often strokes or connected components, into the symbols of the expression. Symbols in the expression need to be labeled with their symbol class, which in some cases might actually depend on more than just the shape of that individual symbol. The parsing of the expression includes looking at pairs of symbols and determines what their relationship to each other is within the expression. With these identified relationships a subset is chosen to represent the entire structure of the expression. All math expression recognition systems have to be able to perform all three of these tasks.

There are currently a number of commercial products which include math expression recognition either as a primary or secondary function. Some examples of products which focus on math expression recognition are InftyReader, PhotoMath, and MyScript Math. These products use the recognition of math expressions in several ways. A common use is for user input into an application, either for display or as type of data to operate on. These applications allow users to input math expressions with mouse, pen, or fingers as a way of dealing with inputting the complex structures of math expressions. On the other side of applications for math expression recognition is extracting expressions from some source. The expressions being extracted can come from written or printed documents that have either been scanned or photographed. The extracted expressions can be used for queries or cross referencing sources.

A problem is not made interesting solely by what it can be used for, but also for the challenge it presents. In general math expression recognition is a hard problem with numerous obstacles. Typeset math recognition shares these problems, along with some problem that are particular to typeset. As discussed above math has a lot of different symbols and a recursive/hierarchical structure. Besides the complication of having a high number of classes the set of math symbols brings several other considerations: number of primitives (strokes or connected components), high intraclass variance, and low interclass variance. The number of primitives will have a large impact on how hard is it to segment symbols in the expression. Luckily many typeset symbols are a single connected component making them easy to segment, but only if they don't contain any noise which might break up the symbol or cause merged components. Depending on how the typeset symbols are created there is the chance that what should be a single connected component is fractured into many. These cases are often hard because the appearance of the fracture might be unique or seen very rarely. Both handwritten and typeset symbols suffer from high intraclass variance. There are uncountable many variations on how to draw a symbol. With typeset a symbol will be very regular within one font, but add in multiple fonts and different styles, including bold and italic, and you start to see a large amount of variance within classes. similarly there is low interclass variance when two different classes closely match. A common example seen all the time is 0 and O. Within symbol classification another set of common mistakes are between capital and lower case letters that have the same shape. The worst instances of this are symbol classes that are exactly the same and are only differentiated by context in the expression. In isolation it would be impossible to distinguish them. An example of this is the horizontal line. A very simple shape which is used all over the place in math: minus, fraction line, hyphen, and overline (bar).

Additional problems which seem particularly evident in typeset mathematical expressions include complexity of expressions, crowding of baselines, and larger sub-baselines. The complexity of expressions for typeset math occurs in two common ways, the number of symbols and the number of levels in the hierarchical structure. These problems naturally lead to the next two. Often typeset math expression are embedded into documents, maybe even inline. To try and cut down on the space taken up by the expressions they are compacted. subscripts and superscripts will often be within the space of their parent baseline instead of offset from it. The increased complexity and higher number of symbols also means more symbols are placed in these sub and super scripts. This means there is more space and other symbols between adjacent symbols on the main baseline.

In trying to classify spatial relations between symbol it is important for the spatial relations between symbols to have high inter-variance and low intra-variance. Unfortunately

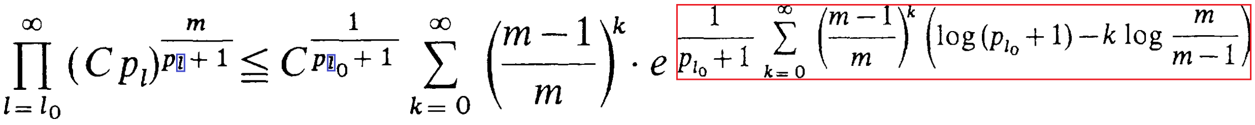


Figure 1.1: A large typeset expression with 83 symbols. The blue boxes show lower case L symbols which have subscript relations to their parent symbols, but are more horizontally adjacent. The Red box shows a superscript baseline for the e symbols to its left. This superscript baseline overlaps its parent baseline.

this is not always the case and one of the reasons parsing mathematical expressions is hard. Common confusions are between right subscript/superscript and horizontal. The main cause of this is low inter variance between the relation classes. A second issue found within the original representation of expressions and the relations between symbols is higher intra-variance within relation classes. There are cases within classes, namely horizontal, which act as outliers in terms of their spatial and visual representation in comparison to the majority of cases for that class.

The cumulative effect of all these challenges makes the problem of typeset mathematical expression recognition hard. The variance approaches taken to tackle this problem have made great attempts to overcome the problems faced. My work aims to show that these problems can be addressed using a combination of simple approaches tailored through careful observation of the problem.

1.1 Approaches to Math Recognition

Mathematical expression recognition is not a new problem and each task required for the recognition has had many approaches tried for solving them. Over the years new approaches and iterations of old approaches as they are revisited have be done. Looking

1.) $$\left(g_{i\bar{j}} + \frac{\partial^2 \phi}{\partial z_i \, \partial \bar{z}_j}\right)$$

2.) $$\leqq \overset{*}{\rho}_{\alpha}(x, y) + \overset{*}{\rho}_{\alpha}(y, z)$$

3.) $$\mathrm{GCD}(\alpha_i, a'_i) = 1, \; i = 1, 2, 3$$

Figure 1.2: (Top) Example expression 1: Notable features are the overline symbols, one of the accent symbol classes. (Middle) Example expression 2: Notable features are the alpha subscript symbols, which are in a more horizontal position, and the fractured symbols, x, y, and comma. (Bottom) Example expression 3: Notable features are subscript i to alpha symbol blocks line of sight to comma and a symbol. There are also a number of baseline punctuation symbols.

at resent work there is still a great deal of variety in the types of approaches. Some of these approaches look at a task or tasks for expression recognition in isolation, such that no information is used from one task for another. Other approaches consider all tasks as a single operation. Another major difference in approaches is the starting data representation used. For handwritten expressions it is common to use strokes, sequences of points recorded in temporal order from pen down to pen up. For typeset the equivalent structure would be connected components. Alternatively the individual points outside of any ordering or grouping could be considered in the same way individual pixels could be considered for typeset images.

A common approach is to use online or offline features with either single classifiers or cascading classifiers for symbol classification. For both segmentation and parsing there needs to be a method by which pairs of components are selected for examination in the algorithm. Frequently used for this operation are time order, linear spatial ordering, or distance measures. The purpose of these is to generate a pool of possible relations from which final relations will be selected by an algorithm to form the recognized expression. The algorithms which are used to select the correct relations, for either segmentation or parsing, have often included grammars or feature based classification. When all tasks for expression recognition are considered at the same time a recent approach is to use neural networks, though they can be used for individual tasks as well. Modern computing has been applying neural networks to more and more problem and recently convolution networks have been applied to typeset mathematical expression recognition [7]. The different approaches to mathematical expression recognition are discussed further in chapter 2.

1.2 Thesis

Our core hypothesis is that a simple recognition system, with basic features and classifiers, can give strong expression recognition rates for typeset mathematical expressions. The basis for this idea comes from previous work [11][6][15] in using the visual density features, geometric features, and contextual features for all parts in math expression recognition, both handwritten and typeset, as discussed in the background chapter. Building on these approaches I have achieved high expression rates for typeset mathematical expressions. Particularly I adapted visual features to improve structural analysis while continuing to use the Hierarchical Contextual Parsing strategy as used by Hu [11]. This is talked about in Section 3.6. The inclusion of contextual features in the system has been shown to be beneficial for all parts [11][15], and I explore better ways to use relevant contextual information. These are the Hypotheses tested:

Hypothesis 1: Parsing can be done effectively using Hierarchical Contextual

Parsing(HCP), with edges weighted by simple classifiers with visual and geometric features that include contextual information from the expression.

Hypothesis 2: The extraction of maximum spanning tree for parsing can be improved by separating the weighting of edges into smaller problems by using multiple classifiers, one binary classifier for horizontal relation classification and another classifier for all other relations.

Hypothesis 3: Contextual features can improve relation classification by identifying baselines of the expression and using them to limit the scope of the context information.

The first hypothesis deals with replicating and adapting the approaches previously used in symbol recognition[6], symbol segmentation, and expression parsing [11] for handwritten mathematical expressions to be used for typeset expressions. My work has shown the approaches to be effective for the typeset expressions. Using the adaptations and extensions for symbol classification, segmentation, and expression parsing I saw recognition rates increase. Symbol classification was able to achieve 99.3% accuracy with correctly segmented symbols. Segmentation gave a symbol recall rate of 99.47%. Parsing detected and correctly classified >99% of relations in expressions. When using all three classifiers within the HCP framework for expression recognition 90.83% of expressions are recognized without error.

Additional contributions include approaches used to alter the representation of math expressions in graphs to better reflect the spatial relations between symbols. These new representations allow relations between the symbols to be more uniform within class, and more distinct between class. Modifications were made to line of sight graph generation to overcome obstacles in typeset expressions and math expressions in general. Finally, I looked at how context information was captured for symbol pair relation classification are tried new methods with a focus on context around baselines.

There are still limitations within the approach used. One is that segmentation of merged connected components is not handled, as these are manually split in the data generation (see Section 3.1). Within the entire data set, 326 cases were found of merged components, out of over 150000 connected components in the dataset, see Section 4.1. In the opposite case, there are connected components which are fractured due to noise in the scanned images. While our system does the segmentation for these cases, it fails for many of them. This is a major source of the remaining errors.

Chapter 2

Background

Mathematical expression recognition is an old problem, typeset expressions were first studied in 1968 by Anderson, R. H. [4], with ongoing and recent attention. Much research has been done to look at the three problems for mathematical expression recognition, symbol segmentation, symbol classification, and expression parsing [21]. Each of these problems hold challenges. In more recent work there have been examples of using a single algorithm, such as neural networks, to do the complete recognition. These systems combine all three parts of the expression recognition into a single operation. Many other systems look at each problem individually, with possibly some information being passed between them. There are a number of ways to approach the task of expression recognition and its sub problems.

The approach examined in this work, as described in Chapter 1, performs the set of tasks for recognition a hierarchical fashion, with segmentation being completed before doing parsing. The symbol classification is done independently from the segmentation and parsing with no information shared between them. This makes the approach less dependent on knowledge of the language itself and instead relies on the visual data being given.

The recognition system implemented and used for this thesis, $Pythagor^m$, was an adaptation and expansion of two prior systems [6][11]. One system was originally for the recognition of symbols in handwritten math expressions [6]. This system was extended to be used with typeset symbol images by using the symbol's contours to represent the symbol with a sequence of points. Additional work was done to extend the symbol recognizer system to include shape context features [15]. The shape context features were used both for symbol features and context features. The shape context features were adapted from the second prior system [11][13][14], designed for symbol segmentation and parsing handwritten math expressions. The two systems had been used in loose cohesion before to do full recognition of handwritten math expressions [11]. The $Pythagor^m$ system used here will be made available as open source.

2.1 Full Recognition Approaches

Algorithms which combine information from each task in their system's decisions have started to be developed more as a result of advancements in machines and algorithms capable of handling complicated tasks. These systems are more often than not built around neural networks. By representing expressions as an image a convolutional neural network can be employed. This allows the lowest level of data, pixels, to be used as input data. A recent implementation of this unified approach was used [7]. This neural network approach directly feeds the expression image into a convolutional network to extract a feature grid learned by the network. The feature grid is passed through an encoder to generate tokens for a markup language. The final step is to give the tokens list to an encoder to build the expression string from the tokens. Using expression images rendered from latex to images their results show their system returning exact image matches of 79.88% [7]. Many system don't use a single algorithm to go from data to a fully recognized expression, instead they look at the problems individually and integrate the solutions from each task.

2.2 Symbol Classification

Symbol classification is an often examined problem in pattern recognition, starting with symbol letter and number recognition and extending to all symbols used in every language[19]. Approaches range from symbol template matching through classification algorithms such as hidden markov models. Features used in classifiers fall into the categories of either on-line or off-line. on-line features are extracted using the knowledge of time order for a set of points that make up a stroke in a symbol. off-line features on the other hand deal strictly with unordered data. In typeset images the data is inherently unorder with respect to time, in both the pixels that make up a symbol and the symbols themselves in the expression. Though there are approaches to that attempt to impose a spatial ordering on expression images, either a linear left to right order or a form of two dimensional ordering.

Some of the common classifier algorithms used with these features for symbol classification are nearest neighbor, support vector machines, random forests, hidden markov models, convolutional neural networks, and bidirectional long short-term memory networks. These classifiers can be used as single instances or combinations of many classifiers used either in parallel or as a cascade[21]. The approaches to symbol classification are many and varied, but for the task of typeset symbol images my work has shown that simple features combined with a simple classifier works well. The classifier used in this work for symbol classification is a random forest. Random forests offer smooth distributions over the feature space and probability confidences in classifications. In running experiments random forests have simple parameter sets and allow for fast parallel training and evaluation. Also the trained random forests are more open to direct examination when trying to understand how the features and classification results are related.

2.3 Segmentation

Segmentation, like parsing, deals with the relationship between multiple components in the expression [5], but has just two cases for classifying the relation between two components. Two components, strokes or connected components, in an expression are either part of the same symbol or they are not. In our system these relations are labeled as Merge and Split. The Merge relation indicates the two components belong to the same symbol in the expression.

Segmentation is a case where an imposed order is sometimes used. Each component checks how well it would merge with the other components next to or near it in the constructed order. This can be done with a dynamic programming approach. One goal of this approach is to reduce the space of possible combinations of components to test for merges. The downside is that linear ordering often isn't a good representation of the complex two dimensional structure of expressions and a good two dimensional ordering is harder to construct. A possible alternative to organizing components into orders it to construct graphs of the components. The graph nodes are the components and the edges represent pairings that should be checked for a merge relation. This graph can be constructed with distance measures, like k-nearest. After determining the space of possible components to merge there needs to be a method for deciding if two components should be merged or split. Merged components are a part of the same symbol. Again this comes down to extracting features and using a classifier. One approach is to use classification scores to see how well the combined components can be classified as a symbol. Alternatively a different set of features can be used to try and capture features representing the merge and split relationship between components.

The segmentation done by this system mirrors [14] and uses a graph based approach, opposed to a cutting method. The graph based approach aims to use a classifier to combine nodes of the graph into symbols. The features used by Hu are visual features, geometric features, and time step features. Similar visual features and geometric features are used with parsing. It was found that a random forest could accurately classify the relations between the strokes as merge or split[11]. In the line of sight graph for connected components 99.77% of edges were correctly classified, see Section 4.3. The initial graph with candidate edges for classification was a line of sight graph. In the line of sight graph only components which could draw a line between their centers without hitting another stroke were included. Symbols were created from the classified graph by combining nodes connected with merge edges. An attempt was made to combine segmentation and parsing into one task with a single classifier, which scored edges in a stroke level expression graph and used Edmonds' algorithm. It was found doing segmentation and parsing separately achieves better results [11]. In $Pythagor^m$ the segmentation is done first to build entire symbols before doing the symbol recognition and expression parsing, giving a hierarchical approach.

2.4 Parsing

There are many approaches to parsing mathematical expressions. Each approach aims to take the components (e.g., points, pixels, strokes, or connected components) of the expression and output the structure of the expression in some form. Different approaches work with different levels of components for the expressions. The expression components given to the parser could be symbols, strokes, connected components, points, or pixels. The $Pythagor^m$ system can use both strokes from handwritten expressions and connected components from typeset images. The last two types of components, points and pixels, are used with complex parsing algorithms. One type of parser that has used these components are convolutional neural networks[7]. In these cases the direct image or vector of points are inputted and the data is already in a structured format. Other parsing methods, which use the higher level components, strokes, connected components, and symbols, require some starting structure from which the final expression structure will be constructed or extracted. Commonly used starting points are graphs, time ordered sequence, left to right ordering, or a form of two dimensional hierarchical ordering. The parsing algorithm might try to find the expression structure incrementally or all at once.

The segmentation and classification of symbols often precede the final task, parsing, of determining the structure of the expression. Using only the spatial relations between components allows the approach used in $Pythagor^m$ to be independent from knowing the component labels and prior knowledge of expression structure. The only information needed for parsing is the visual representation of the expression itself. A common set of features used to represent the spatial relations between components are geometric features. These look at distances and other measures between the points of the components, the centers of the components, or the bounding boxes around the components[3].

Visual features have also been used more recently and have been adapted to include syntactical information with shape context features [3,4], which showed reduced error rates. These features can be used with a graph of symbols in the expression similar to how components were put into graphs for segmentation. Once again their is the option for ordering the symbols by time or space. Either way these features can be used to classify individual relations between symbols and then a set of the relations can be selected to represent the expression structure.

A frequently used technique in mathematical expression parsing, as well as general text parsing, is syntactic methods [2]. One kind of syntactic method used for parsing math expression is grammars. Mathematical expression are a kind of language, so it makes sense to try and capture their structure with grammars. The grammars provide a set of rules for how parts of the expression fit together to form a whole expression. Graph grammars use a graph representation of the expression and use the grammar rules to combine nodes in to a new node. In the case of graph grammars the parsing of the expression is done in a series of steps which continually reduce the graph [10]. This process goes on until a single node, holding a series of rules, represents the entire expression. In most cases the rules used by the grammar for reducing the graph are handmade and it can take many rules to cover the required cases. Even with large lists of rules not all cases maybe covered. The grammar limits the types of expressions that can be recognized to those within the grammar. The alternative to the handmade rules is data driven rules. With data driven rules a training set with ground truths can be used to examine all the cases within the set of training examples. Now this still leaves the limitation of only knowing cases from the training set. To apply the parser to new data a new grammar would have to be extracted.

Another widely used grammar approach for parsing is stochastic context free grammars [2]. Here grammar rules with probabilities are used along with the CYK algorithm to construct the expression. The two dimensional adaptation of stochastic context free grammar also include spatial information for the rules [9]. Once again there is a dependence on the completeness of the rules, which needs to be constructed. Another common dependence with grammars in mathematical expression recognition is the ability to apply a label on each component to determine what kind of symbol in the grammar it represents. Grammars rely on having some prior knowledge of the structures found in all mathematical expressions.

The expression parsing from the segmented symbols in $Pythagor^m$ starts in a similar manner as the segmentation. A line of sight graph was constructed from the symbols and the edges were classified with a spatial relation and given a weight based on the classifiers confidence of the classification. The final set of relations selected from the line of sight graph is done using Edmonds algorithm. Edmonds' finds the maximal spanning tree using those classifier confidence weights on the edges of the graph. Other approaches have also used spanning trees for parsing. Suzuki and Eto used spanning tree extraction over not just relation possibilities in the expression, but also over symbol classes [9]. This approach helped parsing with uncertain symbol classifications. In $Pythagor^m$ for both segmentation and parsing the features used in the classifiers were spatial geometric features and visual density features. The visual density features were used to capture the symbol features, features for the parent and child symbols whose relation is being classified, and contextual features of surrounding symbols.

2.5 Expression Tree Representations

Mathematical expressions are often represented as trees, with symbols or primitives as the nodes in the tree and the edges represent a type of relation between them in the expression. The two common types of trees for math expressions are symbol layout trees and operator trees. The difference between them is what kind of relations you have between the nodes in the tree. An operator tree uses the mathematical operations to structure the tree. The requirement for an operator tree is to have the math operations already detected and identified to build the expression tree. Layout trees instead use the spatial positions of symbols relative to each other to relate them and don't require knowledge of the symbol classes. These relations include relations like superscript and subscript.

2.6 Visual and Contextual Features

Visual features are common for offline classification of symbols, but because they are capable of capturing both shape and spatial information they can be applied to expression segmentation and parsing [11]. shape context features were used to help classify the structural relations between both strokes and symbols[13]. The shape context histogram was placed over the components whose relation was being classified. Each component had its own set of feature values for the histogram. A third set of values were found which represented other components which fell within the space of the histogram. In the end the three separate set of values were found and included in the features. Because the values all come from the same space, the histogram, the spatial information between all three can be related back to each other.

The same technique was used to include syntax information in the features when parsing symbols. Instead of values for parent, child, and other components values were found for letter, number, and other components. Using this syntax information from visual features the expression recognition rate increased. The problem was the usage of syntax information required very accurate labeling of symbols, which was too difficult to achieve with handwritten expressions[11].

Contextual information in visual features has been shown to be beneficial in classification of typeset symbols in my work, in classification of handwritten symbols, and in segmentation and parsing handwritten expressions [11]. The question of how much contextual information is beneficial hasn't been fully explored. My work has shown a very high symbol recognition rate can be achieved with typeset expression images, as high as 99.3% with contextual information in our experiments.

2.7 Summary

In many approaches to expression recognition, the structural recognition depends on the recognition of symbol classes. In some approaches the combined information from segmenting, symbol recognition, and parsing are used together to learn how to structure and label the expressions. Our approach instead does each task individually using the visual representation for features, Section 3.3. The classifiers trained on the features classify the symbol and relation labels in a layout graph, Section 3.1. From labeled and weighted edges of a layout graph a maximum spanning tree is extracted for the expression structure, Section 3.6. The classifying in each task is helped by using contextual information from the expression, see in Chapter 4.

Chapter 3

Methodology

The components of the expression, connected components or symbols, are not represented as their original image from the scanned typeset expression image. Instead the component's image is processed into point sequences using its contours. The raw contours are smoothed and resampled to help reduce noise from printing and scanning the documents. The features for classification of symbols and relations are extracted using these processed contours. The kinds of features used are geometric and visual densities from histograms. Two kinds of histograms are used, two dimensional histograms and shape context features histograms. The visual densities represent several kinds of information: symbol shape, spatial information between two components, and contextual information for either a symbol or component pair.

The hierarchical contextual parsing approach (HCP) uses graphs to represent the expression for segmentation and parsing. The entire structure of the expression can be described by a symbol layout tree (SLT), see Figure 3.1. The nodes of the graph can be either type of component, connected components or symbols. With the relations between the components represented as directed edges in the graph. Classifiers trained on geometric and visual density features are used to classify and score edges in a line of sight graph of the expression components. Segmentation is done by finding nodes connected by edges classified as Merge. The nodes connected by Merge edges are combined into symbols for a symbol level expression graph. The expression at the symbol level is represented as a layout tree, with spatial relations between the symbols. Symbol Recognition does not depend of the structure of the expression once the symbol has been segmented it can be classified. Visual features from the symbol and context from surrounding symbols are used by a classifier to label each symbol.

A part of the examination of symbol layout trees for expressions involved looking at modified representations of layout tree structures. Looking at base punctuation and symbol accents they were previously given horizontal relations with other symbols, but these relations does not resemble other horizontal relations. This can be seen in Figures 3.2 and 3.3. Other expression structures were tried where different relations are used for these symbols. Using the new structures we found the relations for these symbols are more accurately classified, see Chapter 4.

3.1 Data and Representation

The dataset being used is the InftyMCCDB-2, modified from the InftyCDB-2 [20], which contains typeset mathematical expressions from scanned article pages. Initially all the information for expressions is stored jointly between two formats. The visual representation of the expressions are stored as binary images in page files, each containing many different expressions. The ground truth information for all expressions is stored in a single overview file as text. The overview file contains the symbol classification and relation classification between symbols in the expression. The relation classifications represent a spatial relation between the symbols. In the InftyCDB-2 dataset there are seven spatial relations: horizontal

(HORIZONTAL), right superscript (RSUP), right subscript (RSUB), upper (UPPER), under (UNDER), left superscript (LSUP), left subscript (LSUB). An eighth spatial relations was added for baseline punctuation (PUNC), see Subsection 3.1.2.

From its raw format the dataset was converted to a data format to work with the $Pythagor^m$ system and tools[16]. Each expression is still represented in the two separate file types. Each expression is stored in its own pair of files. A PNG image file has the complete visual representation extracted symbol by symbol from the original page image. The symbol classification and relation information for the expression is stored in a label graph file. The label graph file stores a list of components, both connected components and symbols, for the expression along with their classifications. The label graph file also stores the set of directed edges in the symbol layout tree for the expression. The connected components and symbols are linked by id to indicate what connected components belong to which symbol.

3.1.1 Creating Connected Component Ground Truth

The original InftyCDB-2 dataset only contained information for the ground truth expression information at the symbol level. Each symbol was classified and the relations between symbols were given for the expression. The symbols themselves were defined using a bounding box for the page image which contained the expression. No information was given for the connected components of the expressions. The expression images were all separated from page images into their own images, while preserving relative position of the symbols in the expression.

To extract the ground truth connected component information the isolated expression image was examined. First the set of connected components were taken from image using a depth first traversal of pixels and marking foreground pixels within each connected component. A connected component is saved as another bounding box, like the symbols, for its location in the expression image. The set of connected components was then matched to the set of bounding boxes for symbols given in the original ground truth. The connected components were added to the symbol with the smallest bounding box that the connected components bounding box completely fit inside.

There were 326 cases, out of approximately 157700 connected components for around 142100 symbols, where symbols were given no connected components because of merged connected components belonging to two different symbols, affecting <1% of expressions. The merging prevented the combined connected component from fitting into any symbol's bounding box. To deal with this problem, if a symbol was given no connected components it would use its original bounding box as the bounding box of its connected component in the expression image. This would cut the merged portion to only include the part belonging to the symbol. The segmentation done in $Pythagor^m$ system does not handle the splitting of merged connected components or symbols. The connected component bounding boxes are stored in the label graph file and used to extract the connected components from the image during recognition, at which point they are processed (see Subsection 3.1.3).

3.1.2 Symbol Layout Tree (SLT)

Within our system the mathematical expressions are represented using layout trees, with symbols or connected component) as nodes and the relations between them as edges. For the connected components there are only two relations used during segmentation, merge and split. These relations indicate whether two connected components belong to the same

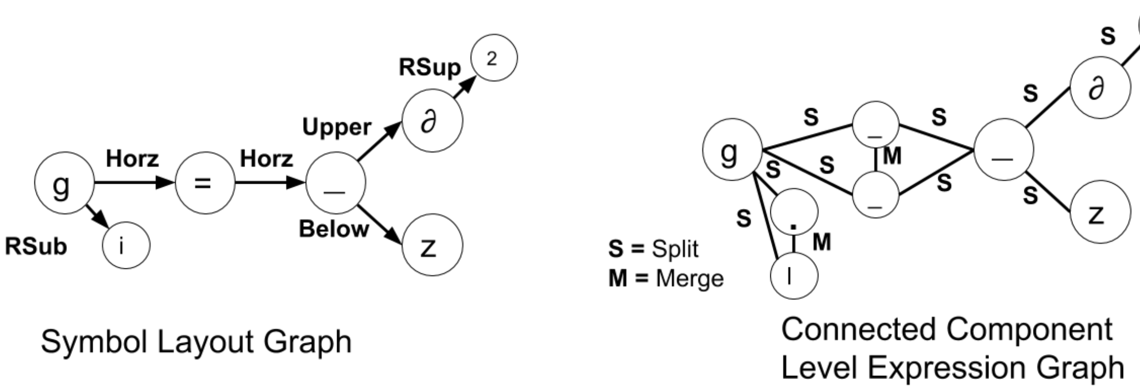


Figure 3.1: (Left) a symbol layout tree with the valid relation classifications on the edges. (Right) A layout tree for the connected components in the same expression with edges labeled as merge or split. Merge edges indicate the connected components belong to the same symbol.

symbol or not. For symbols the spatial relations are used, see Figure 3.1. For classification of these relations spatial and visual features are used, see 3.3. There are a few particular cases where the layout structure being used causes problems with parsing expressions.

Two prominent instances of problems in layout structures can be found for the horizontal spatial relationship, baseline punctuation and accent symbols. These two cases are fairly similar to each other, in that both have horizontal relations which don't line up with most other horizontal relations. baseline punctuation, commas, periods, and ldots, are much lower than both their parent and child symbols in the symbol layout tree. Accents, overlines, tilde, hats, dots, vectors, and checks, are all found above a symbol or set of symbols but are considered a part of the main baseline instead of their own above the contained symbols. For reference see Figures 3.2 and 3.3. Example expression 3 has several commas with horizontal relations to the symbols before and after them in the expression. Accents can also have superscript and subscript relations. This is another issue with the accents. The child relations are from the accent but are usually place in reference to the symbol under the accent. In example expression 1 there are two overlines, one is above a z and has a subscript relation to the j symbol. While its possible for a classifier to learn these outlier relations, the final parsing algorithm will still end up having to choose between these outliers and the incorrect relations which look more like other horizontal relations. With the case in example expression 1 of a j subscript to the overline there is the chance the j will be parsed as a subscript of the z instead. This can also lead to baseline punctuation becoming a part of the subscript baseline or horizontal relations going to the symbol under the accent instead of the accent itself. An arguably better approach to the outlier relations is to change the expression representation in the symbol layout tree structure to better match the expected spatial and visual layout of symbols. Again see Figures 3.2 and 3.3.

Punctuation Relation. These baseline punctuation relations have more in common with subscript symbols than other horizontal relation symbols. Having the baseline punctuation separate from the main baseline and moved to its own sub baseline from its parent symbol allows for its horizontal relation to be replaced with a more typical representation of horizontal relationships. Baseline punctuation is given its own baseline with its own new relation class punctuation (PUNC). The separate baseline allows for easily reintegrating parsed expressions into the original expression structure with a simple graph transform without any additional information, like symbol class.

Baseline punctuation in some cases has another obstacle, subscripts. When a symbol has both subscripts and punctuation there is even more confusion in parsing. From the very start there is a chance that the subscript will block the line of sight between parent symbol and the punctuation, see Figure 3.7. This problem with line of sight is handled in Subsection 3.2.1. The next challenge is dealing with the fact that the subscript and punctuation look horizontal to each other. This is often reflected in the final parsing, with the punctuation having the subscript symbol as its parent with a horizontal relation. The current PUNC relation does not directly handle this problem, but because the PUNC relation is specific to a particular set of symbol classes it seems like there should be a way avoid these errors. The immediate response is to use symbol classes to have the classifier learn these cases. Instead experiments, see the symbol shape context experiments in 4.5.1, show using features for the child's shape help in these cases.

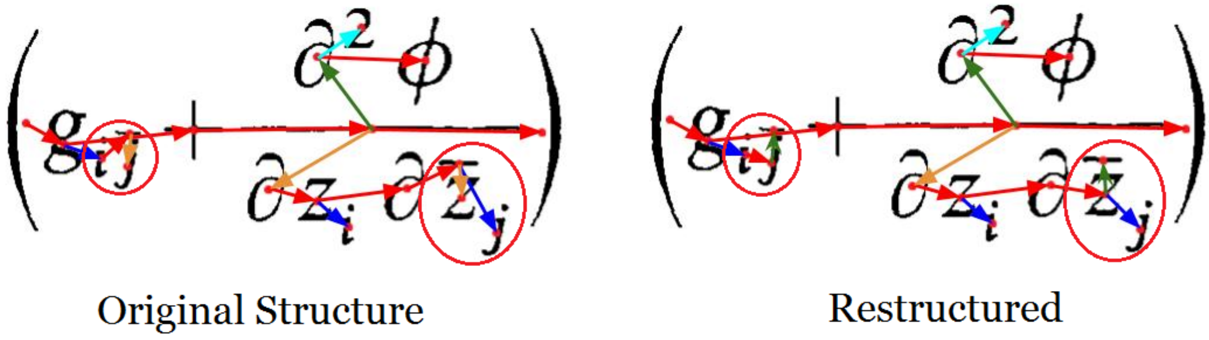
Accent Restructuring. Using the same approach of creating a new relation and baseline does not neatly transfer to the problem of accents. Like baseline punctuation there is the problem of the incorrect horizontal relations being added to the expression. For accents this incorrect relation is between the parent symbol to the accent and the symbol below the accent. For reference see Figure 3.2 and its overline symbols. In example expression 1 there is a z symbol with an overline, a relation between the previous symbol and the z more closely match other horizontal relations. This relation will have a stronger confidence than the relation from the parent symbol to the accent in many cases. In the restructured layout trees this relation is made the correct horizontal relation in the ground truth representation of the expression. The accent is given an above relation to symbol under it, again see Figure 3.2. This does mean in the cases where there are multiple symbols under the accent the range of the accent will have to be recovered after parsing. Any child relations of the accent are given to the symbol, or right most symbol, under it.

Removing Trailing Punctuation. The removing of trailing punctuation in expressions was done as part of the processing of the expressions from the InftyCDB-2 dataset for any punctuation at the end of expressions. The expressions came from scanned mathematics papers and for expression that were placed inline the trailing punctuation, commas or periods, was often included. The trailing punctuation itself is not really a part of the mathematical expression. For many applications of typeset math expression recognition including the trailing would be a hindrance. In order to better represent the problem the trailing punctuation was removed.

The trailing punctuation in expression was removed by finding the symbol with the right most bounding box right side and checking its class to see if it was a comma or period. All commas and periods found at the end of expressions were removed. This change affected 20% of expressions. Many of the expression with trailing punctuation were smaller, as they were the ones included inline in the papers.

3.1.3 Image Processing

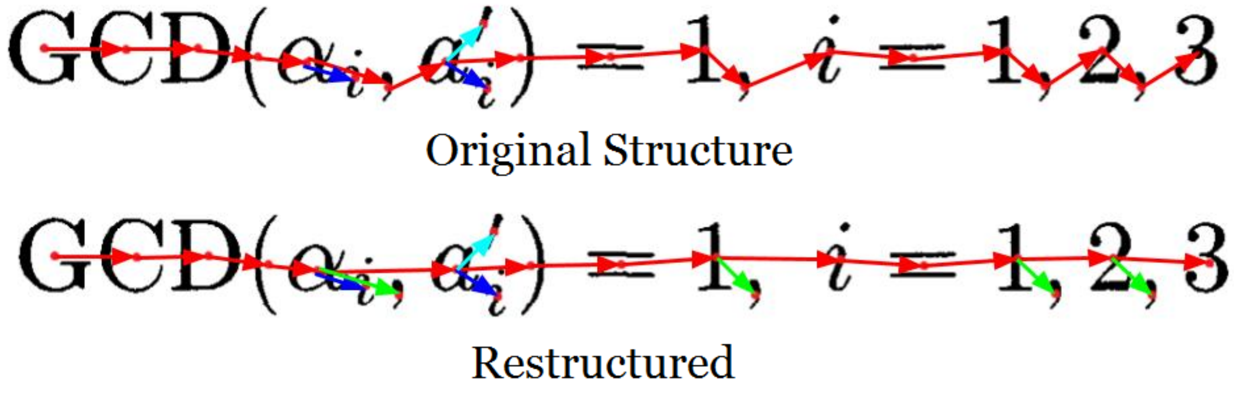
As stated in the previous section, the input data is images of the mathematical expressions. Each symbol in the expression is made up of either a single connected component or a set of connected components. Before trying to do any kind of recognition the symbols are modified to be represented by sequences of points. To change the symbols from being images to point sequences the contours are taken for each connected component in the symbol. The contours of a connected component consists of a single outer contour and any number of inner contours. The extraction of contours simply takes all the edge points of the connected components and choosing a arbitrary starting point travels along the edge adding the pixel locations to point sequences. This is done for each edge of the connected component.



Original Structure

Restructured

Figure 3.2: Graph Structures with arrow from parent to child: Red-Horizontal, Dark Blue-Subscript, Light Blue-Superscript, Dark Green-Upper, Orange-Under, Light Green-PUNC (Left) Example expression 1 with original ground truth expression structure. The baseline punctuation is part of the main baseline and has a horizontal relation to its parent and child. Overlines are a part of the main baseline and child has Under relation. (Right) Example expression 1 altered expression structure. Baseline punctuation has its own baseline and has PUNC relation with parent. accents are now in Upper baseline. Child symbols of accents now belong to the symbol below them.



Original Structure

Restructured

Figure 3.3: Graph Structures with arrow from parent to child: Red-Horizontal, Dark Blue-Subscript, Light Blue-Superscript, Dark Green-Upper, Orange-Under, Light Green-PUNC (Top) Example expression 3 original ground truth expression structure. The baseline punctuation is part of the main baseline and has a horizontal relation to its parent and child.(Bottom) Example expression 3 altered expression structure. Baseline punctuation has its own baseline and has PUNC relation with parent.

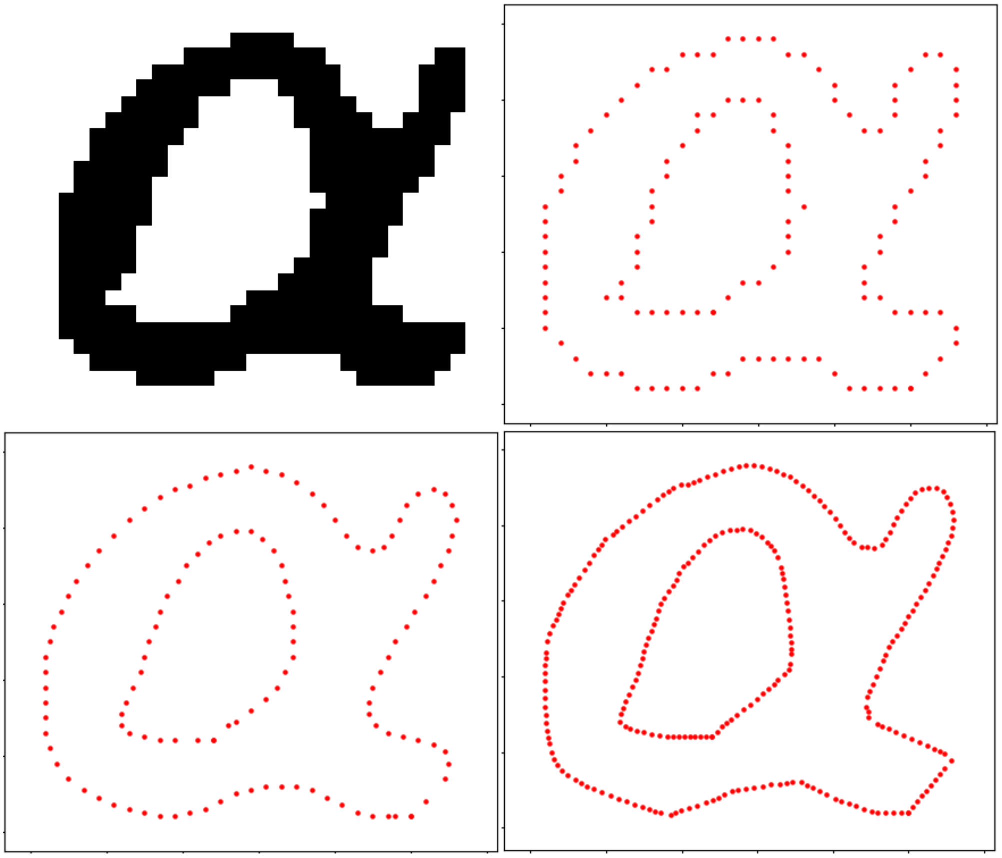


Figure 3.4: A processed alpha symbol from example expression 2. (Top Left) The original symbol image. (Top Right) The extracted contour of the symbol image. (Bottom Left) The smoothed contour using smoothing distance of 2 to average point locations with points before and after in sequence. The smoothed contour has the same number of points. (Bottom Right) The symbol contours after using Catmull to do more smoothing and resample the contours.

The contours are able to capture the symbol shape with less information than the entire image of the symbol. Due to directly taking the contours from the symbol image there is noise from the scanned image. The later preprocessing by the $Pythagor^m$ system for point sequences, used for both typeset data and handwritten strokes, does not completely handle this noise. So, an added smoothing step is used in the contour extraction for typeset images. This smoothing replaces a point in the sequence with a point whose position is determined by taking the mean position of the original point and a set number of points before and after the original in the sequence. Previous work found using a distance of two, the previous two points and following two points, for the smoothing improves symbol recognition the most. Using a higher smoothing distance resulted in losing shape characteristics of the symbols. Curves and corners of the symbols became straighter and less distinct. This smoothing is done on each contour for a connected component. The smoothed contours have the same number of points as the original and the same end points.

We next apply the preprocessing steps described in Davila et al. [6] for handwritten symbol recognition. this resamples the point sequences using Catmull-Rom splines. This resampling adds additional points the points sequences to make the distance between points more uniform.

3.2 Expression Graphs

Throughout the stages of expression recognition the expressions are represented as graphs. There are two kinds of expression graphs that are used, primitive level graph and symbol level graph. The difference between these two type of graphs is what the nodes in the graph represent. In a primitive level expression graph each node represents a connected

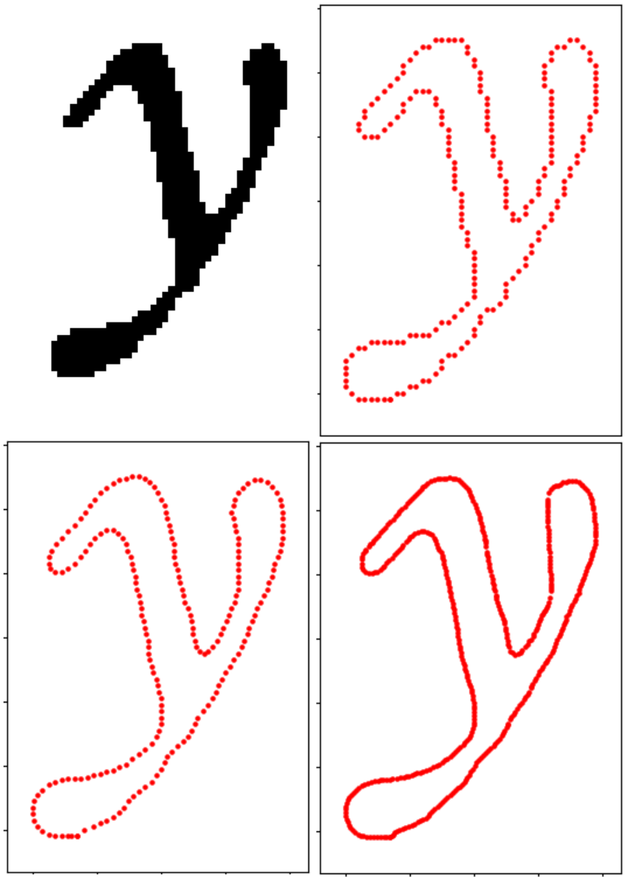


Figure 3.5: A processed y symbol from example expression 2. (Top Left) The original symbol image. (Top Right) The extracted contour of the symbol image. (Bottom Left) The smoothed contour using smoothing distance of 2 to average point locations with points before and after in sequence. The smoothed contour has the same number of points. (Bottom Right) The symbol contours after using Catmull to do more smoothing and resample the contours.

component in the expression. The edges between the nodes represent the relations between the connected components. This relation can either be one of the spatial relations or it can be labeled with Merge or Split, see Figure 3.1. The merge relation indicates these two connected components belong to the same symbol in the expression. Split simply indicates the opposite. For the $Pythagor^m$ recognition system used the primitive level graphs are only used for segmentation and only use the merge/split relations for them.

The symbol level expression graphs instead have each node represent a symbol from the math expression. The edges between the nodes have one of the spatial relations or a label of NoRelation. The NoRelation label is used to either indicate that the edge has not been classified or the edge is not in the valid ground truth expression graph. The expression graph representing the structure of a mathematical expression will be a single direction spanning tree.

In both primitive and symbol graphs the edges of the expression graph are directed, though all merge and split edges in the primitive expression graphs can be taken as reflexive. This means for spatial relation there will be a parent and a child node or symbol. The spatial relation describes the relation from the parent to the child. For example if there is a superscript relation between two symbols the parent symbol is the base symbol and the child is the superscript symbol. With all ground truth horizontal relations the parent is to the left of the child, see Figures 3.2 and 3.3.

3.2.1 Line of Sight Graphs

Initially when the structure of the expression is not known and has to be parsed there needs to be an initial set possible edges between the nodes of the expression graph. A fully connected graph would ensure that no edge from the ground truth graph is absent from this starting graph, but would also include a large number of false edges which are not in the ground truth graph. An alternative approach is to use a line of sight graph. A line of sight graph adds edges between two components, connected components or symbols, which can "see" each other in some geometric space and are not block by other components. While this does not guarantee all the edges from the ground truth will be included, it reduces the number of false edges. The typeset symbol graphs in the training set had a precision of 5% with fully the connected graph approach, or 1:19 ratio ground truth edges to false edges. The line of sight graphs had precision levels around 20%, or 1 to 4 ratio of ground truth to false edges. The recall of the line of sight was over 99% and with some modification to find punctuation reached >99.8% recall for ground truth edges.

The Line of Sight graphs are constructed with the method outlined in [11]. This method uses the center of a component's bounding box as an eye and checks the angle ranges blocked, determined by angle between eye and points on the convex hull, by other components. For each component node in the graph it checks the angle ranges of all other components in order of smallest point to point distance between itself, parent node, and the other component, child node. The parent node starts by being able to see in a full 360 degree range around its eye. For each child node its blocking angle range is compared against the still yet unblocked angles of the parent. If there is an overlap between the child's blocking angles and the still viewable angles of the parent node then it is determined that the parent can see the child and the edge is added to the line of sight graph. Then the seen child node's angle range is removed from the set of angles viewable from the parent's eye. The line of sight graph is also constructed to be reflexiv. If a parent can see the child an edge is also

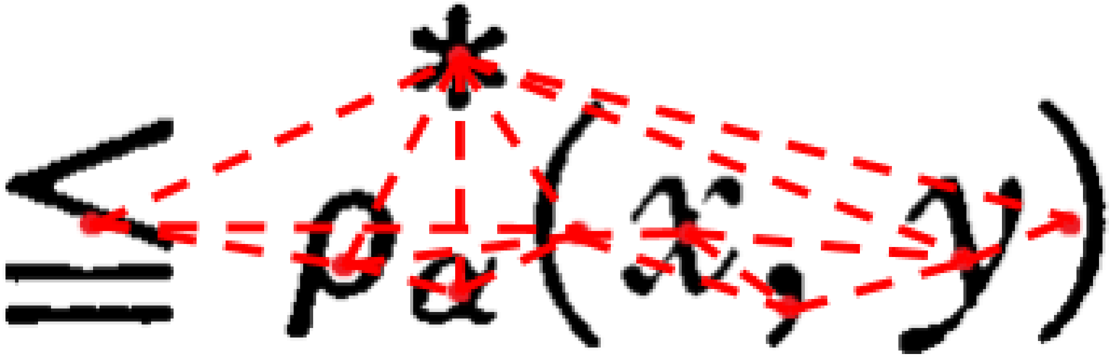


Figure 3.6: The dotted lines show edges between symbols in a line of sight graph from a segment of the example expression 2. These edges will be scored and classified during parsing then Edmonds algorithm will select a maximum spanning tree from these edges.

added from the child to the parent, see Figure 3.6.

Using this approach for line of sight graph building gives a recall just over 99% for the edges belonging to the ground truth expression graph and a precision of 22%. A few of the stated difficulties with typeset mathematical expressions directly impact the line of sight graph building. Crowded baselines and extended sub-baselines result in obstructed line of sight between horizontal symbols on the same baseline, see Figure 1.1.

Punctuation Finding. A common problem is for baseline punctuation, commas, periods, and ldots, to be blocked by subscripts. On inspection of theses cases of blocked line of sight it can be seen that for many of them no straight line can connect these two symbols without passing through another blocking symbol, see Figure 3.7. This indicates that the some variations for line of sight wouldn't be as beneficial in trying to capture these missing cases. Some variations [11] make use of multiple eyes for the parent symbol, alternate representations of the child symbol, or modified rules for when angle ranges are considered blocked. The multiple eyes could include points on the convex hull, the corners of a bounding box, or eyes at different heights along the center line. With no line between any point on the parent to the child multiple eyes would have any effect in these cases. The same can be said of different representations of the child symbol. The third type of variation allows line of sight through at least one symbol which would normally block the line of sight. This could potentially allow the missing edges to be found, but would greatly reduce the precision of the line of sight graph.

The problem with the third approach is that it allows too much transparency. For the case of baseline punctuation there are two characteristics which allow for high recall in recovering missing edges to them, location and size. Baseline punctuation are most commonly found to the lower right of their parent symbol and are usually several times smaller. I modified the line of sight to check if a blocked symbol is contained in this location and much smaller than itself. If both cases are satisfied then an edge is added from the parent to blocked child. To restrict where baseline punctuation was looked for I used an angle range of 310-360 degrees in which the child symbol must be contained. The constraint on size compared the area of the parent to child and required the parent to be atleast 3.5 times larger. See Figure 3.8. While there is the potential for edge cases where these do not hold the majority of baseline punctuation in the inftyCDB-2 dataset missed by normal line of sight are recovered using this method. This increases the recall of correct relations by .33% in the line of sight graph. There is a reduction in precision when this method is used, from both small, non-punctuation symbols being seen in the same angle range and large parent symbols seeing many other symbols which would normally be blocked. Depending of the size

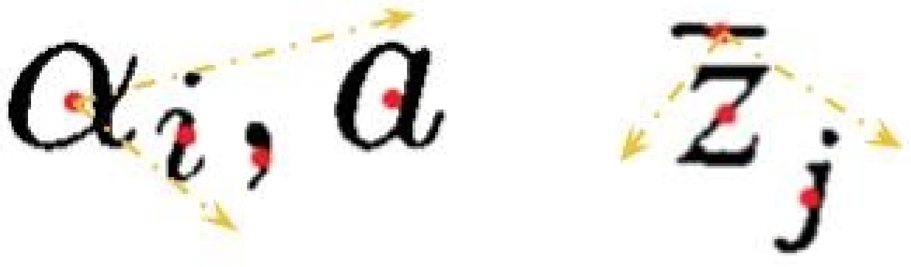


Figure 3.7: Examples of blocked line of sight. (Left) The subscript i symbol is blocking the line of sight from the alpha to both the comma and a symbols. Because the i symbol is closer the angle range it blocks is removed from the visible range before the alpha looks to the comma or a. (Right) The z symbol blocks line of sight between the overline and the j symbol. In the original graph structure that 'j' would be a subscript to the overline accent. After restructuring the j is a subscript to the z instead.

and angle range allowed for the finding the punctuation the precision can drop to 18-19% precision, with an increase of recall to 99.74-99.84%.

Shrinking Blocked Range. The second modification looks to reduce the amount of blocking caused by the subscript and superscript symbols along the horizontal baseline. To do this a transparent end technique is used, where the range of angle blocked by a child symbol are reduced by shrinking the angle range size of the child symbol. Before removing the child's angle range from the still visible range of the parent it is shrunk by a percentage factor based on the original size of the angle range. An equal portion is removed from the top and bottom of the blocking range. This means superscript and subscript symbols will have a small portion of their blocking range remain visible at the top and bottom of the symbol. This approach manages to increase recall of the line of sight, but with another reduction to precision, to below 16% precision from 19% precision.

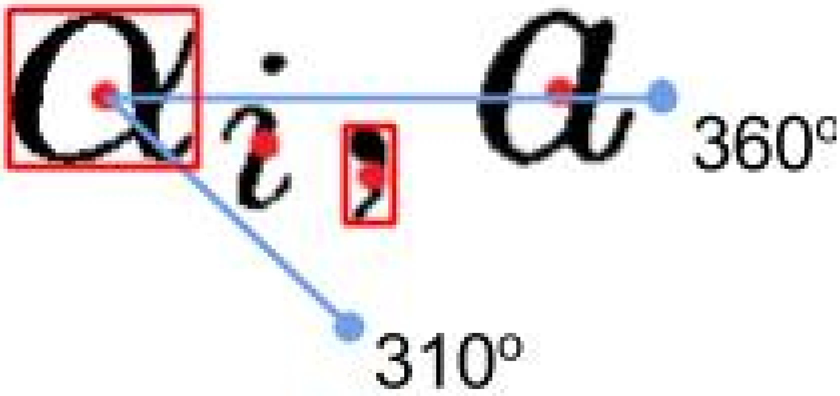


Figure 3.8: Example of finding blocked punctuation symbols from example expression 3. The parent symbol alpha is looking for small symbols in the 310-360 degree range from its center eye. The comma symbol fits in that range. The area of the alpha symbols bounding box will be compared to the area of the comma's bounding box. If the parent's bounding box is 3.5 times larger in area an edge will be added to the line of sight graph from parent to child.

3.3 Features for Symbol and Relation Classification

Principle to the methods explored in this thesis is the types of features used for each step in the expression recognition. Only two types of features are used, spatial and visual. The spatial features include geometric feature sets and is used with pairs of connected components for segmenting and pairs of symbols for parsing. Visual density features are captured using two kinds of histograms, shape context features and 2D Grid Histograms. These histograms are able to capture features in a few different ways. Like geometric features the histograms can examine pairs of expression components, but can also look at individual components. The individual component features can be used in symbol classification, but they have also been found useful in parsing. The methods of feature extraction mentioned have dealt with directly observing what is being classified. For symbol classification histograms of the just symbol are used. For parsing the histograms can look at the parent, child, and space between them. There is another form of features which are indirectly related to whats being classified, contextual features. Contextual features, unlike component or component pair features, don't explicitly measure what being classified. The contextual features gather information from around the subject, either locally or globally. The histograms are used to gather local contextual features in the form of visual density of all components not being classified in the same area. The amount of context is controlled by the size of the histogram being used. These contextual visual densities are useful for all tasks in math expression recognition, see Chapter 4.

3.3.1 Geometric Features

Geometric features are used for connected component pairs and symbol pairs. The geometric features are measures in distances, area overlaps and differences, size ratios, and angles. For pairs of connected components or symbols a common set of features are distances and differences based on the bounding boxes around each stroke/symbol. These distances include: distance between center points, difference in vertical position of bounding box tops and bottoms, difference in horizontal positions of left and right edges, difference in area, and amount of overlapping area.

Another set of geometric differences looks at the set of points in each stroke/symbol. Features of this nature are maximum and minimum distance between the two set of points, A measure of how parallel the two set of points are, distance between the mass centers, and difference in average vertical position and horizontal position.

Geometric features for symbols can also look at specific points in each stroke/symbol, namely start and end points. The start and end points between the two sets of points can be used in another set of distances, horizontal differences, vertical differences, and angles between them. With expression images and typeset symbol there are no real start and endpoint to the stroke or symbol. Artificial start and end points could be given, but they might not hold the same level of information. Geometric features work well when trying to discern the spatial relationship between symbols, see experiments using only geometric features for parsing in Section 4.5.1.

3.3.2 Shape Context Features

The shape context features [3] are circular histograms constructed from concentric circles and lines passing through the center to divide the space into uniform angular sectors. The cells or bins of the histogram are defined by arcs from the concentric circles and segments of the lines passing through the center of the circles, See Figure 3.9. This figure shows a shape context features histogram being used for shape density features. Shape context feature circles are centered on the center of the bounding box containing the symbol or component pair. Shape context features are used to represent visual density features of a symbol have a outer circle with a radius equal to the distance between the center to the furthest most point of the symbol or component pair. The concentric circles' radii are uniformly distributed between the center point and the outer circle's edge. The angular sectors are evenly distributed at angles starting from 0 degrees from the horizontal. With contextual shape context the radius of the outer circle uses the radius of the base symbol shape context multiplied by the radius factor. The radius factor controls how much of the surrounding contextual information will be included in the contextual features. Figure 3.10 shows contextual features for a overline symbol in example expression 1. The shape context has a radius factor of 8, meaning the entire shape context radius is 8 times the distance between the overline's center and outer most point.

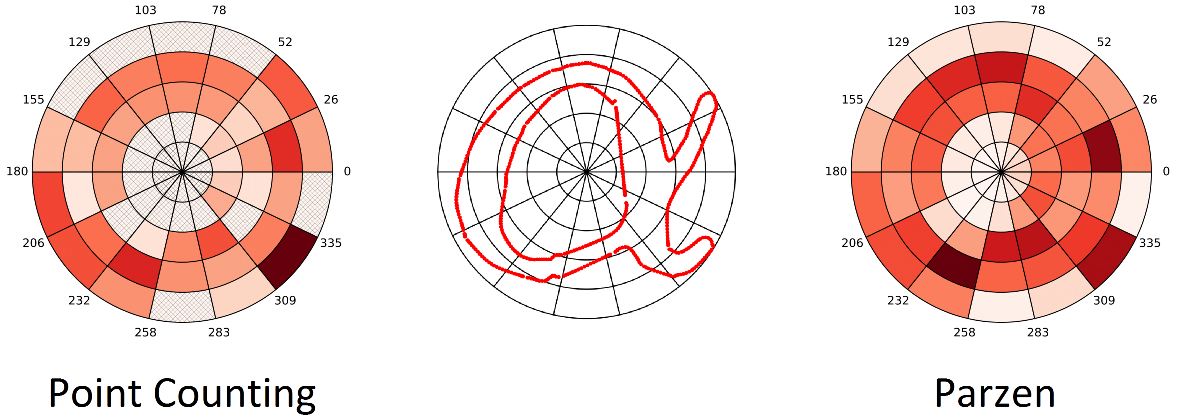
The simplest approach to measuring visual density with the shape context feature is to count how many points from the symbols fall within each bin and using these values. The problem with this is the discrete nature of point counting does not handle variance well. If a line moves from being along one side of a bin edge to another the two bin values shift dramatically. An approach to help deal with variance is using Parzen shape context features.

With Parzen a distance measure is used between each symbol point and the centers of the bins. The distance measure comes from a Gaussian pulse centered at the symbol point. The symbol point can now contribute to every bin value proportional to its distance. The Parzen method requires a sigma value used in the Gaussian pulses from each point. The sigma is derived from the maximum radius of the shape context times the standard deviation factor. The sigma factor is used to control how flat or peaked each Gaussian distribution is for the symbol points. A large sigma factor will result in a flatter distribution, thus a more even distribution of value among all the bins.

The approach used for shape context in parsing and segmentation is to use a single shape context and collect three sets of values for the bins. One set where the bin values come from the points in the parent, one set for the child points, and one set for all other points within the shape context's range. When doing symbol classification two sets of values can be used for the symbol and all surrounding context points. The multiple set of values from the same histogram allows spatial information from each set to be directly compared to each other. In classifying relations this information can be used to tell where the parent, child, and context are relative to each other.

3.3.3 2D Grid Histograms

The grid histogram can be viewed as a grid of cells or a grid of points that make up the corners of the cells. The method of grid histogram used in this work has a grid of corner points. The number of points in the grid is used as a measure of its resolution. As more points are placed in the same area the resolution of measured features increases. For symbol and symbol context features the histogram is centered at the center of the symbol's bounding



Point Counting

Parzen

Figure 3.9: shape context features (SCF) histogram for capturing shape density features of alpha symbol in example expression 3. The SCF has 5 circles, 14 angle sectors, and radius factor of 1 (sized to fit from symbol center to outer most point). (Left) Point counting method is used for bin values and is normalized by total bin values. The hashed bins have a value of 0. (Center) symbol points overlayed on SCF. (Right) Parzen method is used for bin values. This gives smoother density with smaller differences between neighboring bins.

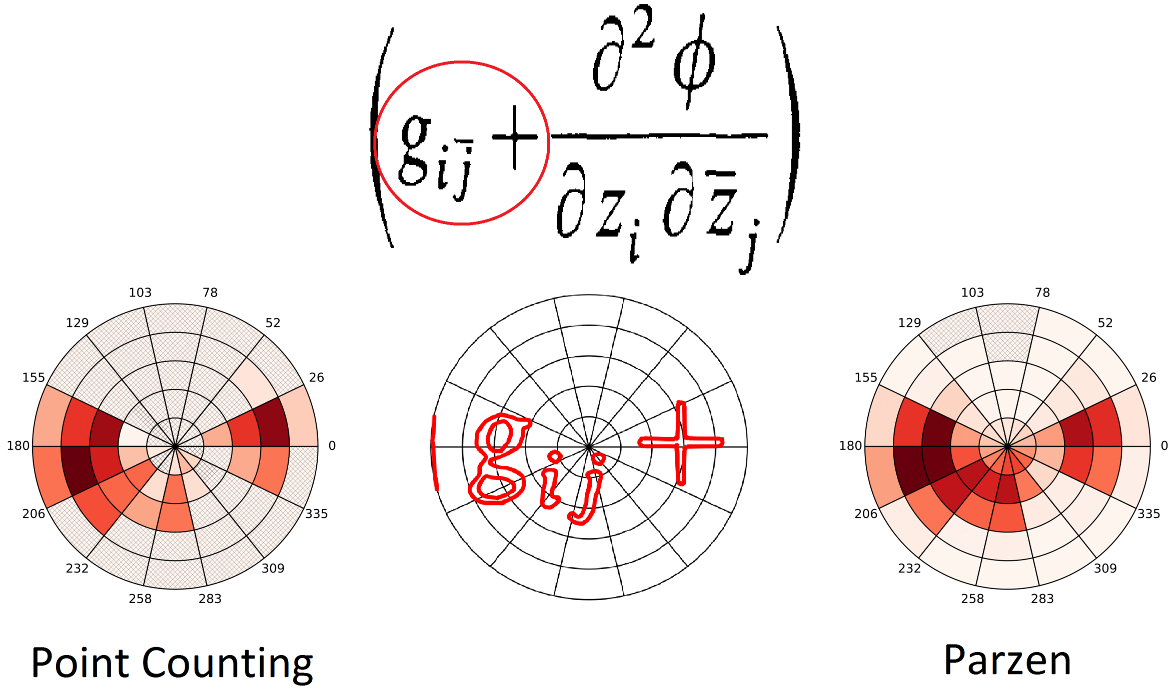


Figure 3.10: shape context features (SCF) histogram for capturing context density features of overline symbol in example expression 1. The SCF has 5 circles, 14 angle sectors, and radius factor of 8 (radius is 8 times larger than needed to fit entire overline symbol. Center of SCF is the center of the overline symbol being classified. (Left) Point counting method is used for bin values and is normalized by total bin values. The hashed bins have a value of 0. (Center) context points overlayed on SCF. (Right) Parzen method is used for bin values. This gives smoother density with smaller differences between neighboring bins.

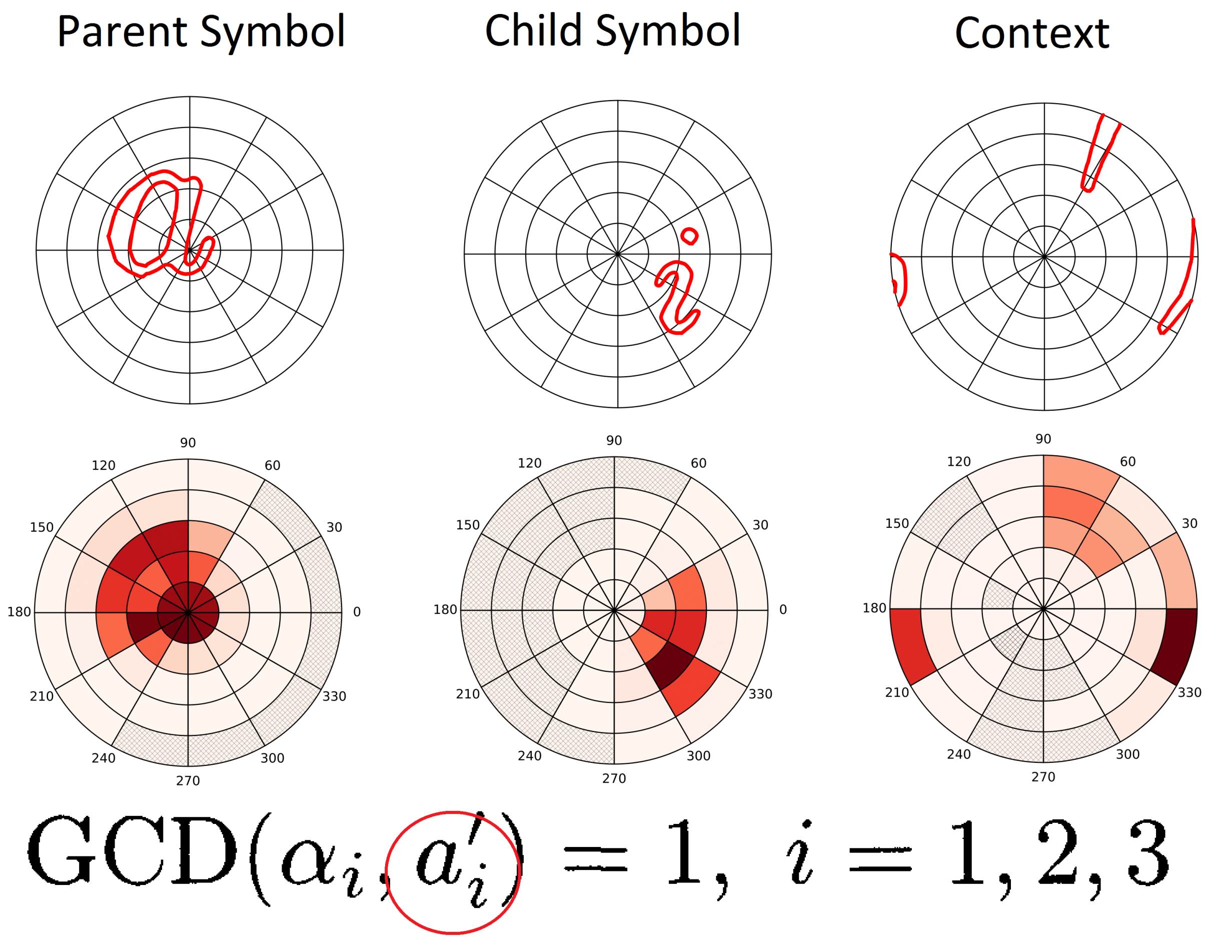


Figure 3.11: shape context features (SCF) histogram for capturing spatial density features of the relation between a(parent) and i(child) symbol in example expression 3. The SCF has 6 circles, 12 angle sectors, and radius factor of 1.5 (radius is 1.5 times larger than need to fit both parent and child symbol in SCF). (Left) Symbol points and feature values for parent spatial density. (Middle) Symbol points and feature values for child spatial density. (Right) Context points and feature values for context spatial density.

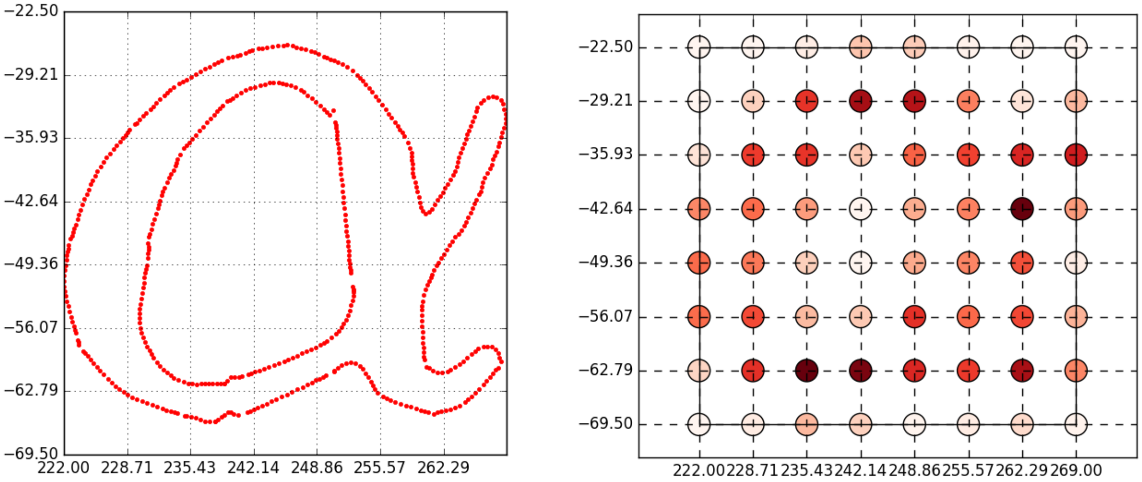


Figure 3.12: (Left) alpha symbol points overlayed on 2D Grid Histogram (Right) 8x8 2D Grid Histogram using fuzzy point histogram for capturing symbol shape density features. Alpha symbol taken from example expression 3.

box. The longest dimension of the symbol's bounding box is used as the side length of the entire grid. A similar method is used for arranging the grid around a pair of components, but with the bounding box containing both components instead. For capturing context, either for a symbol or pair of connected components, the histogram is centered again on the symbol or component pair. The size of the histogram can be increased by a factor of the base length calculated for symbol or component pair features to include more context.

The feature values for the histogram are based on the distribution of points relative to the corner points. This can be done in a number of ways with different distance metrics. The method used for symbol classification was a fuzzy histogram [6] were each symbol point contributed to the four nearest corners (the corners of the cell it is in) based on the euclidean distance between them. Another approach is to use Parzen based contribution from each

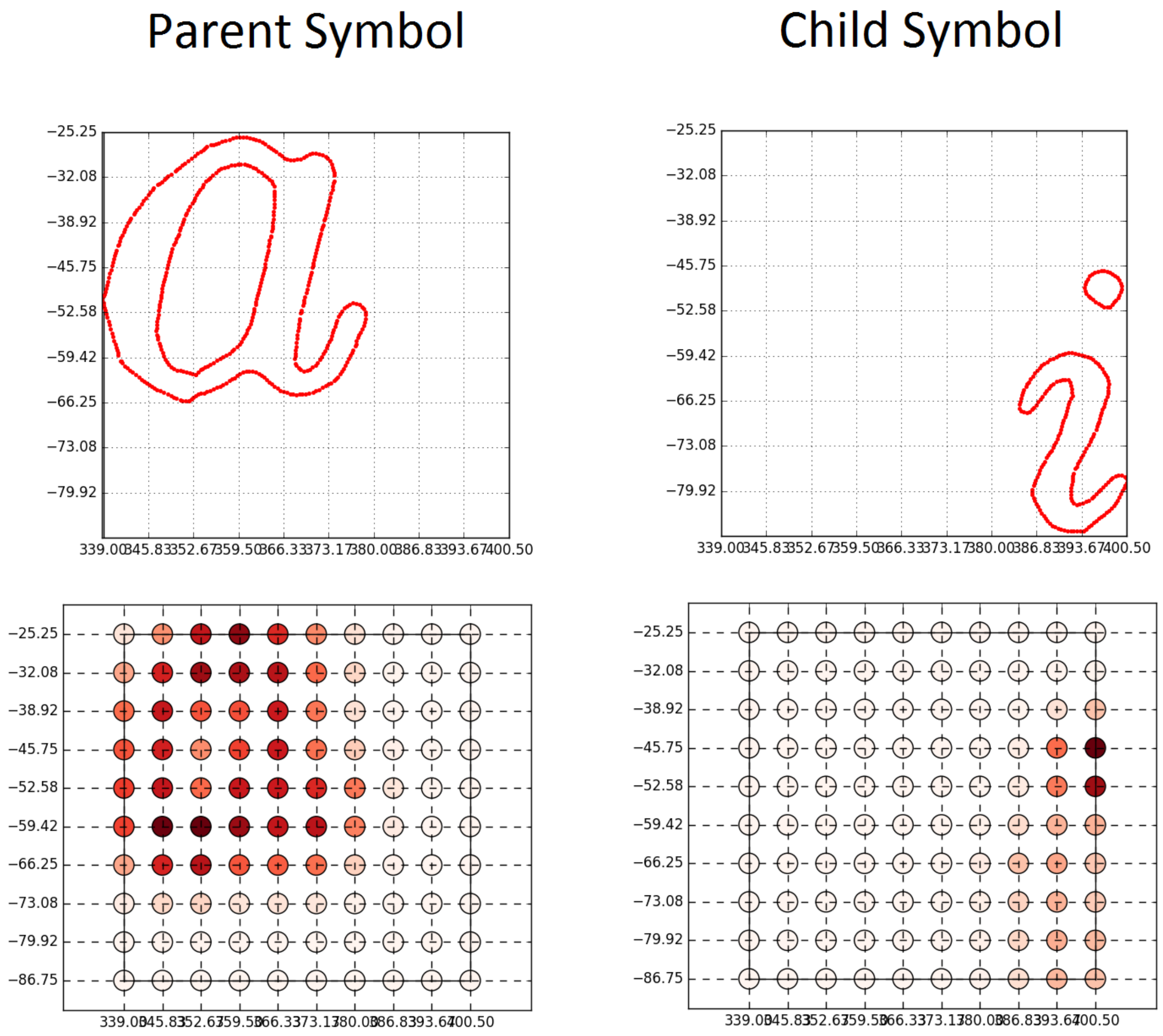


Figure 3.13: 2D Grid histogram spatial densities for a relation between parent symbol a and child symbol i from example expression 3. Grid is centered between the parent and child symbols and is sized to fit both symbols. (Left) Parent symbol points in grid histogram and the density features. (Right) child symbol points in grid histogram and the density features.

symbol point to corner. The Parzen method has the same benefits as it did with shape context features, see 3.3.2. The Parzen method simulates contribution of each symbol point using a Gaussian pulse. This allows a symbol point to contribute to all corners in the grid, but will contribute much more to those closer to it. This gives a smoother distribution of density and helps with variance in the positions of symbols. The sigma for the Gaussian distributions is the product of the grid's diagonal length and a sigma factor. The sigma factor is used to control how flat or peaked each Gaussian distribution is for the symbol points. A large sigma factor will result in a flatter distribution, thus a more even distribution of value among all the corners.

3.3.4 Maximum Spanning Tree Context

The maximum spanning tree (MST) context features[11] are used to directly visual represent relations between symbols and capture visual density features using histograms. A two dimensional grid histogram was used for capturing the visual density features. The way the relations are represented visually is a line of points from the geometric center of the parent symbol to the center of the child symbol. The number of points is directly related to the distance between the two centers.

This feature is used to capture the information presented by doing an initial pass with a classifier. The classifier will give the class probabilities for an edge and these will be used as class confidences for the edge when extracting mst context. A histogram is centered on a relation between two symbols and will record the visual densities for each visualized mst edge. Just like normal relation visual densities multiple sets of values can be recorded for the histogram, but instead of values for the parent, child, and context, it can be values for particular relation classes. For example if the intent was to get the horizontal relations visual densities the histogram can record the visual densities of all the visualized relations and each point in the relation lines will be modified by its horizontal class confidence. This way edges that have a higher probability of being horizontal relations contribute more to the histogram. Doing this gives a visual density representation of the surrounding relations to try and help in classifying other relations.

In testing this feature was used to visualize possible baselines, horizontal relations, visual densities as another form of contextual information. The results showed a decrease in accuracy when this feature was used. More experimentation will have to be done to get a better idea of how mst context can be used.

Experimental Results. The results did not show MST context helping parsing. An initial pass of parsing was done using a classifier trained with just the geometric and normal grid histogram features to classify the set of possible relations in the line of sight graph. Next the MST context features came from these classified relations and a second pass of classifying all the relations in the line of sight graph was done. At the end Edmond's algorithm found the maximum spanning tree for the final expression layout structure. The MST context extracted only looked at the relations classified as horizontal, to capture a visual representation of the expression's baselines. Three methods were used for the initial parsing pass. The first approach kept all relations from the line of sight graph and used all of them for the MST context. The second approach used Edmonds' algorithm to get a maximum spanning tree and only used the relations in the mst for the MST context. The final approach also used Edmonds' to extract the MST in the first pass, but only after applying the unique symbol relation constraint, see Section 3.6. The unique symbol relation constraint forces each parent node to have a most one child node with a given spatial relation. So, each symbol can only have one horizontal child, one superscript child, one subscript child, etc. A grid histogram using parzen was used for the MST context features, with a resolution of 10x10 and size to fit the parent and child symbols. None of the tested methods seemed to show improvements in relation classification or parsing the expression structure.

MST context caused relations to be skewed towards horizontal relations in all the tested cases. It maybe that MST context does not work well for parsing as it is too simple to capture the intended information. The MST context was meant to represent both the information gained in the first pass of relation classification with the remembered confidences and it was suppose to add information relating the position of symbols to the baseline structure of the expression. This seems like a lot of information to be conveyed by the single line drawn between symbols. There may still be a better way of representing this information visually.

3.3.5 Directionally Extended Context

Previous experiments had found increasing the size of histograms to capture more context information from surrounding symbols can help to an extent in parsing typeset expressions. Directionally extended context looks at increasing the amount of context by looking along the horizontal axis, as most expressions have a more horizontal than vertical structure. Two forms of directionally expanding histograms were tested, one for grid histograms and another for shape context histograms. For grid histograms additional columns can be added to the grid to extend it horizontally. For shape context the radius was increased and a bounding box was used to restrict the points which contribute the feature values. The directionally extended context features were combined with the geometric features and additional visual density features. For the extended grid histogram it captured parent, child, and the context visual densities. For the extended shape context only context features were captured and the other visual densities were from a grid histogram. Neither of these extended context showed improvements over the normal increased in size density histograms.

The extended context features also failed to show any improvements during cross validation. The two dimensional grid histogram has repeatedly shown worse performance when larger histograms are used to capture more context. My speculation on this is either it is an issue with high variance of context further out from the relation being classified or the searches in my experiments need to be expanded to include more combinations of parameters for the features. For shape context histogram the improvement in relation classification matched normally expanding the shape context to include more context. Restricting the points considered to those horizontal with the parent and child symbol points didn't help. It may be that the other points aren't as noisy as thought. Another reasonable explanation has to do with the effect of adding more visual density features. As more histograms are added or histograms are made higher resolution with more features values the total number of feature values goes up, but the number of helpful features does not increase as quickly. One way to improve the extended shape context is to instead of removing certain context points, remove certain bins in areas which have less or no useful information. This will hopefully improve the ratio of helpful to unhelpful feature value.

3.4 Symbol Segmentation

The algorithm used for segmentation follows the binary approach used previously for segmenting handwritten symbols using visual features [11]. A primitive level expression graph, with each connected component as node, is used to represent the entire expression. The initial graph structure is set by building a line of sight graph between the connected components. A classifier is used on each edge in the graph, using features from the connected component pair. The edges in the primitive expression graph are directed but edges are added reflexively to the line of sight graph. This means each neighboring connected components in the graph will get features for both parent child directions. The edge is given a classification of either merge or split by the classifier. A sub-graph is extracted with all the nodes from the original graph and only the edges which were classified as merge. Connected nodes in the sub-graph will be merged into symbols. All nodes that can reach each other are made into a single symbol node for a symbol level expression graph.

3.5 Symbol Recognition

Symbol recognition is done from a symbol level expression graph. The edges in the graph are not considered for this task. Symbol recognition takes each node in the symbol graph and extracts the features for the symbol and uses a classifier to label the node with one of the symbol classes. The features used are visual densities for the symbol shape and local context.

3.6 Expression Parsing

Parsing is done on the symbol level line of sight graph. Unlike segmentation the classifier for parsing does not directly classify each edge. Instead the classifier is used to give each edge probability scores for each of the possible spatial relations, including NoRelation. From these scores the highest is selected to give the edge a weight and possible classification. NoRelation is never used as a possible classification, instead if NoRelation is the highest score the next highest is selected. If this was not done when a max spanning tree is found it might select a NoRelation edge with strong confidence, there are many of these in each line of sight graph. This would result in an invalid final expression structure. With the weighted expression graph Edmonds' algorithm[8] is used to select a maximum spanning tree and resolve any cycles. The maximum spanning tree is used as the final expression structure.

3.6.1 Cascade Classifiers

The purpose of using cascade classifies for parsing is to use multiple stages of classification and parsing to iteratively improve the results. The cascade designed for parsing the typeset expression uses two classifiers and operates in two stages. Both classifiers use the same features and the same training set. The difference between them is how the training samples are labeled and the classifications they can give. In the typeset expression more than 70% of the relations are classified as horizontal. This leads to the majority of errors made when parsing involve the horizontal relation class. Common errors include false edges with the ground truth relation of NoRelation being classified as horizontal, the reverse where horizontal relations are not included in the final expression structure, and confusions between RSUB and horizontal relations.

The first phase of the cascade looks at detecting the horizontal relations in the expression. The classifier used for this is trained on three classes instead of all the relations. The three relations are Horizontal, NotHorizontal, and NoRelation. The difference between NotHorizontal and NoRelation classifications is that NotHorizontal are relations found in the ground truth expression graphs, but are not horizontal relations. NoRelation are still any relation that doesn't show up in the ground truth expression graph. In the first stage this classifier is used to classify all the edges in the line of sight graph as horizontal or NotHorizontal and give a score to those classified as horizontal.

The second phase classifies and scores all the remaining edges in the expression graph without scores. The classifier for the second phase is trained using all the relation classes. This phase behaves just like the normal parsing for all the unclassified edges, giving each edge a weight equal to the highest probability from the classifier. Once all the edges have a probable relation class and a weight, from the probability score of one of the two classifiers, Edmonds' would be used to extract a maximum spanning tree.

3.6.2 Unique Symbol Relation Constraint

When using Edmonds' algorithm a maximum spanning tree is selected without any consideration of the possible classes of the relations, only the weight of the edges are considered. This means the final set of selected edges might have an invalid mathematical expression structure. A common example is having two horizontal relations from a single parent symbol. In Figure 3.14 you can see an possible scored expression graph before Edmonds' has been run. The rho symbol has two horizontal relations to its children, one to the alpha and one to the left parenthesis. In this case the relation to the alpha should be a subscript, but because it looks more horizontal that relation gets a higher probability score. This is actually a common problem in relations. One possible way of handling this problem is to use a unique symbol relation constraint which will prevent invalid expression structure, like multiple horizontal relations. The hope is the unique symbol relation constraint will choose the correct relation to keep, thus filtering invalid relationship hypotheses.

The unique symbol relation constraint implemented looked for all cases where a parent symbol has multiple edges with the same relation to child symbols. The relation with the highest score for that relation remains. All the duplicates would be forced to chose their next highest relation score. If the next most likely relation is NoRelation then the relation with the next highest probability is used. If an edge ever has a score of 0 it was removed. The removal does run the risk of no valid maximum spanning tree being found by Edmonds' algorithm. Using this constraint helps with cases of RSUB relations being classified as Horizontal. There was a small decrease in detected correct relations, due to constraint selecting the wrong relation, but overall the expression rate went up, see Tables 4.3 and 4.4.

3.7 Evaluation Metrics and Tools

The evaluation of results is done using the LgEval library[17][16] created for the analysis of expression graphs represented as layout trees. The layout trees examined can have primitives, connected components for typeset, or symbols as the nodes in the graph. These tools produce evaluation metrics and for both the primitive and symbol level. These tools also produce confusion histograms for error analysis.

The evaluation metrics are based on recall and precision of labels in the produced layout trees. These labels include labels on the nodes for symbol classification and labels

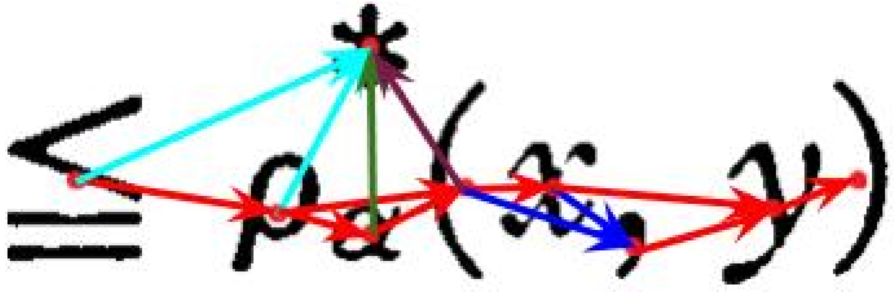


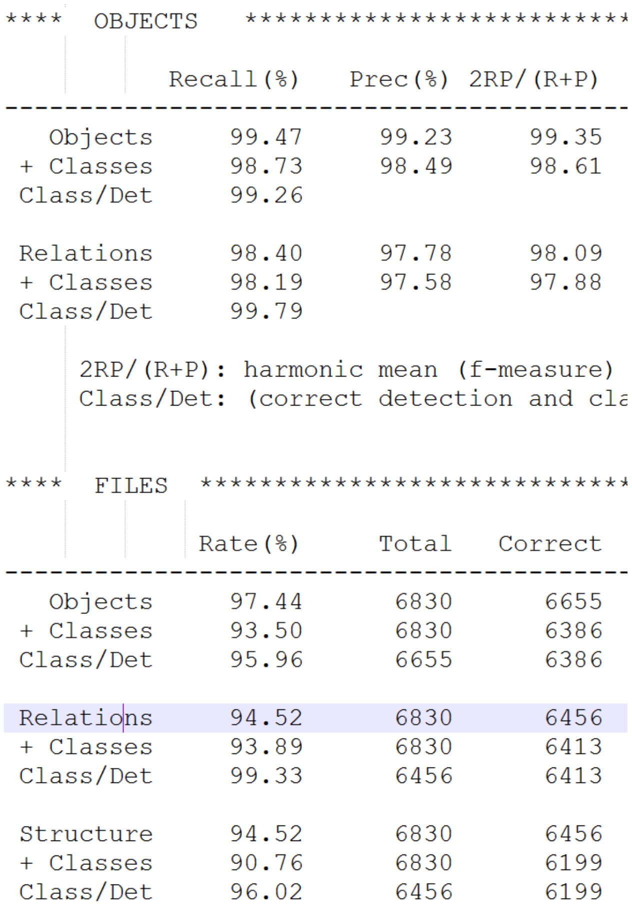
Figure 3.14: Graph Structures with arrow from parent to child: Red-Horizontal, Dark Blue-Subscript, Light Blue-Superscript, Dark Green-Upper, Orange-Under, Light Green-PUNC. A partial line of sight graph (reduced number of edges for clarity) with many of the more confidant edges with most likely relation classification. This Graph would be passed to Edmonds' algorithm to find maximum spanning tree. Notice rho symbol has two likely horizontal relations to child symbols. Both can be selected by Edmonds', but this would form an invalid expression structure.

on the edges for relations between the nodes. The recall indicates what percentage of the correct labels are present in the produced tree out of all the correct labels in the valid layout trees. Precision indicates what percentage of the produced labels are valid labels. Using these many different metrics can be examined. The primary metrics are symbol detection, symbol classification, relation detection, and relation classification, see Figure 3.15. symbols are segmented correctly when all the correct primitives for a symbol are labeled as being part of the same symbol. symbol classification requires the symbol to be detected correctly and have the right symbol class label. relation detection indicates whether an edge exists between the two components in both the produced tree and the valid layout tree. Relation classification depends on the correct detection of an edge and the correct relation label for the edge. The symbol detection is used for segmentation analysis. symbol classification is used for symbol recognition. Expression parsing uses both relation detection and classification.

Another important aspect of evaluating results is error analysis. The confusion histograms are a good tool for seeing what mistakes are being made. Besides the standard table confusion histogram for both symbol labels and relation labels, there are subgraph confusion histograms. These allow detailed case analysis to be viewed in an organized way. With a subgraph of size one the confusion histogram would show errors in single nodes in the symbol layout tree. This would contain segmentation errors for a symbol and classification errors. If the subgraph was expanded to size two this would be errors between two nodes, and include relation parsing errors. This tool also allows for the reporting of which expressions these errors are occurring in, see Figure 3.16. These tools provide informative and valuable insight to the behavior of our recognition system.

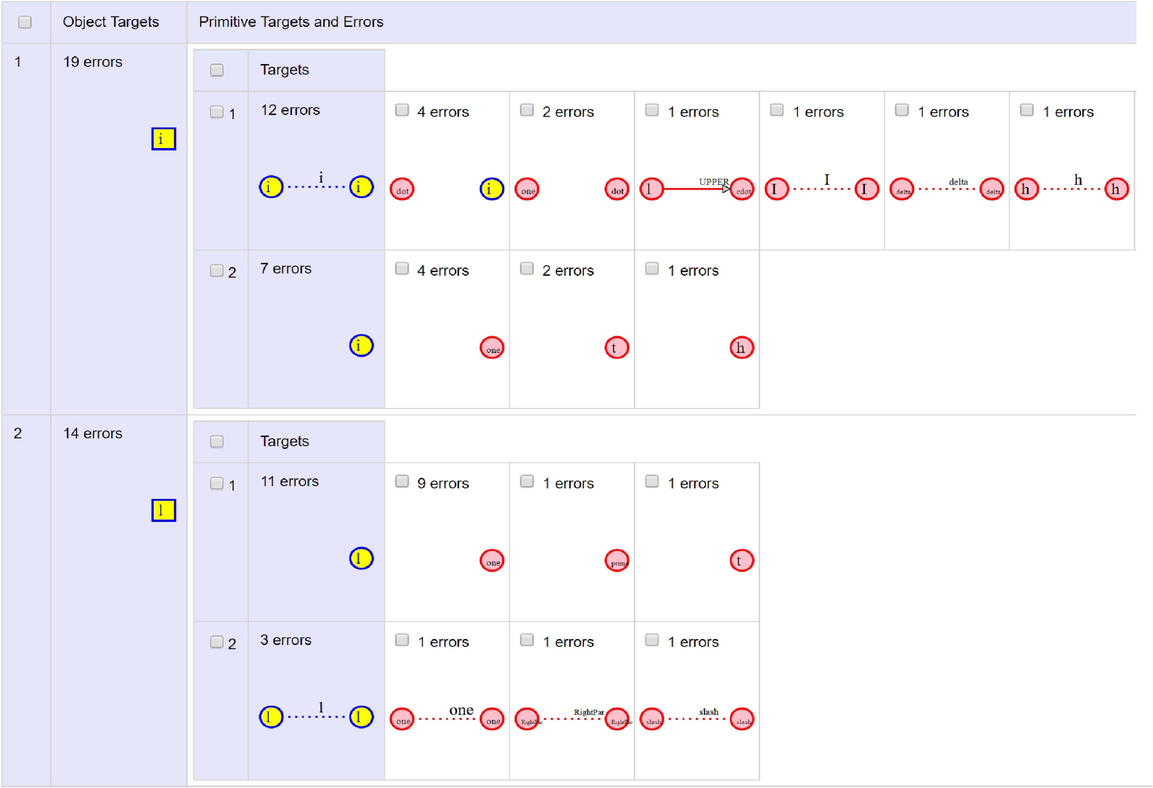
3.8 Summary

The recognition of scanned typeset math expressions from the InftyMCCDB-2 dataset, created from the InftyCDB-2 dataset, is done by first processing the connect components from images to point sequence representations. The expression are represented as symbol layout trees with an initial edge set from doing line of sight[11]. Multiple forms of the layout trees were tried for parsing symbols. The connected components are segmented into symbols by classifying the relations between them in the graph as either merge or split. All connected components connected by merge edges are combined into a symbol. At the symbol level another line of sight graph is constructed, but with the modification of looking for block punctuation. Two set of classifications are done independently. The edges between symbols are given relation classifications and weights from one classifier and nodes are given symbol



```
****  OBJECTS   **************************

           Recall(%)   Prec(%) 2RP/(R+P)
------------------------------------------
   Objects     99.47     99.23     99.35
 + Classes     98.73     98.49     98.61
 Class/Det     99.26

 Relations     98.40     97.78     98.09
 + Classes     98.19     97.58     97.88
 Class/Det     99.79

     2RP/(R+P): harmonic mean (f-measure)
     Class/Det: (correct detection and cla


****  FILES  *****************************

             Rate(%)     Total   Correct
------------------------------------------
   Objects     97.44      6830      6655
 + Classes     93.50      6830      6386
 Class/Det     95.96      6655      6386

 Relations     94.52      6830      6456
 + Classes     93.89      6830      6413
 Class/Det     99.33      6456      6413

 Structure     94.52      6830      6456
 + Classes     90.76      6830      6199
 Class/Det     96.02      6456      6199
```

Figure 3.15: This is a portion of the evaluation output showing the symbol level metrics at the top (labeled as Objects) and expression rates at the bottom (labeled Files).



□ | Object Targets | Primitive Targets and Errors

1 | 19 errors

□ Targets

□ 1 | 12 errors | □ 4 errors | □ 2 errors | □ 1 errors | □ 1 errors | □ 1 errors | □ 1 errors

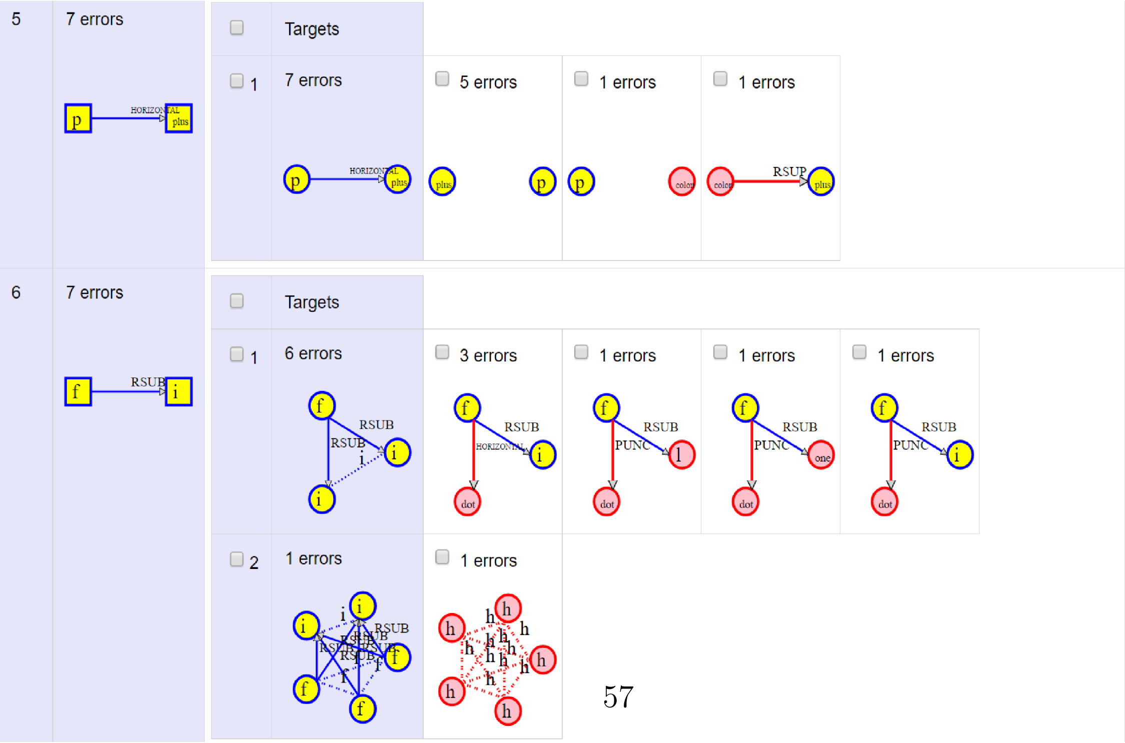
□ 2 | 7 errors | □ 4 errors | □ 2 errors | □ 1 errors

2 | 14 errors

□ Targets

□ 1 | 11 errors | □ 9 errors | □ 1 errors | □ 1 errors

□ 2 | 3 errors | □ 1 errors | □ 1 errors | □ 1 errors



5 | 7 errors

□ Targets

□ 1 | 7 errors | □ 5 errors | □ 1 errors | □ 1 errors

6 | 7 errors

□ Targets

□ 1 | 6 errors | □ 3 errors | □ 1 errors | □ 1 errors | □ 1 errors

□ 2 | 1 errors | □ 1 errors

Figure 3.16: (Top) shows samples of confusion histogram output for subgraphs of size 1. The first symbol is an i and is segmented incorrectly and miss classified. (Bottom) shows samples of confusion histogram output for subgraphs of size 2. The first error shows relation detection errors between p and +.

classifications from a separate classifier. In all three sets of classifications, connected component relations, symbol relations, and symbol class the features used are spatial features and visual density features. The relations use the geometric features, which have been expanded from 38 [11] features to 50. All three type of classifications use the visual densities from histogram. These visual densities capture symbol shape, spatial information between pairs of components, and contextual information. The histograms used were two dimensional grid histograms and shape context features histograms.

Chapter 4

Results and Discussion

A series of experiments were conducted to examine the features and approaches to symbol segmentation, symbol recognition, and expression parsing. The dataset used is the InftyMCCDB-2 as described in Section 3.1. Each recognition task was first explored individually and final testing was done by doing full expression recognition from the connected component level.

The experiments look at feature parameters for adapting the geometric and density histogram features used, adding new contextual features, alternative expression layout structures, and modified processes for parsing. The feature parameter experiments are done through cross validation testing on the training set and look at the direct recall rates of the classifiers. The assumption for segmentation and parsing is that a classifier with better recall in classifying the relations found in the line of sight graphs will do better in segmentation and parsing.

The results of these experiments shows the hierarchical contextual parsing approach is effective in recognizing scanned typeset math expressions, see Tables 4.3 and 4.4. This recognition of expressions is aided by the contextual features (3.3), the altered layouts (3.1), and a unique symbol relation constraint (3.6).

4.1 Data

The dataset used for all experiments and testing is the InftyMCCDB-2 dataset created from the InftyCDB-2 dataset, see Section 3.1. From the dataset 19381 Typeset math expressions are split into a training and testing set. The expressions come from scanned English mathematics papers. The dataset contains a total of over 142000 symbols instances from 213 different symbol classes. The expressions range in size from a single symbol to more than 75 symbols, with an average of 7.33 symbols per expression. The symbols themselves are made up an average of 1.11 connected components, with a standard deviation of .36. This puts the large majority of symbols at having only one connected component. There are several symbol classes with a standard representation containing two or three connected components: i, j, equals, leq, geq, double prime, Theta, colon, cdots, ldots, plus minus. Many other instances of symbol instances with multiple connected components are fractured symbols whose connected components were split by printing and scanning noise. In extreme cases a symbol with a normal representation of a single connected component can be fractured into more than ten connected components. The original dataset does not deal with symbols at the level of connected components, this was added for this thesis and is outlined in the data processing section in the methodology chapter.

To create the training and testing sets the original set of expression was divide into two sets of 12551 and 6830 expression with approximately the same distribution of symbol classes and relation classes. The splitting of data using a sum of squared differences only concerned the class distributions and not the sizes of the expressions. As a result the average number of symbols in expressions is slightly higher in the training dataset than the testing by .6 symbols per expression, but the training dataset also have a higher standard deviation in the number of symbols per expression.

During the course of the investigations for parsing typeset math expressions several alternate structures for expressions were examined, see Subsection 3.1.2. This involved rewriting the ground truth representation of the expression to match the new structure. These changes did not effect any symbol classification or segmentation information. Here is a list of the expression set versions and the changes made from the original expression structure:

- **Expressions v1:** No Changes
- **Expressions v1b:** Modified v1. Trailing punctuation removed from end of expressions.
- **Expressions v2:** Modified v1b. Switched accents (tilde, hat, dot, check, and vec) from being in baseline.
- **Expressions v3:** Modified v2. Added PUNC relation for baseline punctuation (comma, period, ldots).
- **Expressions v4:** Modified v3. Added ACCENT relation for accent overline symbols,
- **Expressions v5:** Modified v3. Switched overline symbols same as other accents.

Table 4.1: Common Symbols in Dataset

Relation Frequencies		
Symbol Class	Training	Testing
Right Parenthesis	6.5%	6.5%
Left Parenthesis	6.4%	6.4%
One	4.5%	4.5%
Comma	3.8%	3.7%
Equal	3.6%	3.6%
two	2.8%	2.8%
Zero	2.7%	2.7%
i	2.0%	2.0%
Minus	1.9%	1.9%
Plus	1.8%	1.8%

Table 4.2: Relation Frequencies in dataset

Relation Class Frequencies				
Symbol Class	v1b Training	v1b Testing	v5 Training	v5 Testing
HORIZONTAL	77.7%	77.4%	73.5%	73.3%
RSUB	11.9%	12.0%	11.9%	12.0%
RSUP	6.2%	6.2%	6.2%	6.2%
PUNC	–%	–%	4.2%	4.2%
UNDER	3.2%	3.2%	1.4%	1.4%
UPPER	1.0%	1.1%	2.8%	2.9%
LSUP	<.1%	<.1%	<.1%	<.1%
LSUB	0.0%	0.0%	0.0%	0.0%

4.2 Classifiers

For all experiments and testing random forests classifiers are used [18]. The implementation used for the random forests is from the scikit-learn library for python. This selects random features from the pool of all features to use in building the decision trees. All comparisons of results use consistent set of parameters for the random forests. The initial set of parameters used for the random forest included: 50 trees, max depth of 40, max number of selected features was the square root of the total number of features, and Gini measure is used for making the splits in the tree. These parameters come from previous work using a similar set of features [11] and the same approach to segmenting and parsing handwritten expressions. After some initial testing the max number of features was changed to a constant value of 30. This was done after observing lowered results when using higher resolution visual density histograms and include a higher number of such histograms. The reasoning for this is that visual density histograms contain many feature values to describe the entire space, but at time many of these feature values are not discriminative enough to help in classification. At higher resolutions and more histograms the ratio of discriminative features to those not helping in classification become worse. The trade off is the possibility of over fitting to the dataset. This is handled by using five fold cross validation on the training set for feature parameter selection.

4.3 Symbol Segmentation

The experimentation done with segmentation was limited to adapting the previous approach tried for binary segmentation of math symbols in handwritten expressions using the spatial and visual features [11] outline in methodology and adapting the visual density features for the typeset data. The final results using geometric features, visual densities, and multi scale context, using multiple histograms of different sizes for contextual information, achieved a 99.47% symbol recall on the test set from InftyMCCDB-2 of over 44,700 symbols, comprised of 49,600 connected components. This includes the 326 cases of manually split merged connected components.

Segmentation experiments deal exclusively with which features set should be used and what parameters are optimal for these features. The experiments are done by doing searches over the feature parameters in cross validation on the training set. The metric used evaluation in the cross validation is classification rate of relations between connected components, either merge or split. The merge relation indicates that these two connected components belong to the same symbol. The parameters which give the best results in cross validation are used in testing on the test set, where symbol recall is used for evaluation. Symbol recall measures how many symbols from the testing set were able to be fully constructed with all connected components from the segmentation.

4.3.1 Symbol Segmentation Experiments

The main experiments consisted of looking for optimal resolution the visual densities, comparing the use of shape context feature densities with 2 dimensional grid histograms, and finding the best amount of context to use for segmentation. Two grid searches were done with five fold cross validation on the training dataset, see Section 1.1 in Appendix. In the grid search what is reported are the classification rates of the edges in the graphs being examined. While this does not directly indicate the trend in symbol recovery for the parameters being tested, it is reasonable to believe the parameters which produce a classifier with more accurate classifications would give higher symbol segmentation rates. Each grid search paired one of the visual density methods with the geometric features, those used in [11]. The resolution of the visual densities was varied. Also in the grid searches the amount of context viewed was increased by increasing the size of the histogram. For both types of histograms using a single histogram was used to collect all three sets of values, parent density, child density, and context density. This means increasing the size of the histogram while keeping the same absolute resolution, number of values in the histogram, lowers relative resolution in regard to the symbols being viewed.

The first set of experiments yielded the result that both shape context features histograms and 2 dimensional grid histograms do equally well when sized to only fit the parent and child components, with relation classification rate around 99.74%. Also in terms of most effective resolution for the histograms it seemed that lower resolution histograms did better than higher. In this area a more expansive grid search over even smaller resolution histograms might yield better results. The lowest shape context resolution, 3 circles and 6 sector angles, and the lowest grid histogram, 5 by 5 grid, did the best. In these experiments as the size and amount of context viewed increased the shape context feature histograms did better than the grid histograms, by ¡.1% in relation classification. This is possible due to lower variance in the shape context histograms. This is what lead to choosing the shape context histogram for the multi scale context features.

The next set of experiments looked at using multi scale context. These experiments added a second density histogram to the previously done grid searches, were the second histogram was a shape context feature histogram which only looked at context. Now the set of features consisted of the geometric features, a histogram to get parent, child, and context with a base size, and a shape context histogram with a radius factor greater than 1.0 to get only context. The resolution of the of the context only shape context feature histogram was kept constant, 3 circles and 10 angle bins, based off results of the previous experiments [15]. Also the parameters of the other histogram were kept constant. The only thing being varied

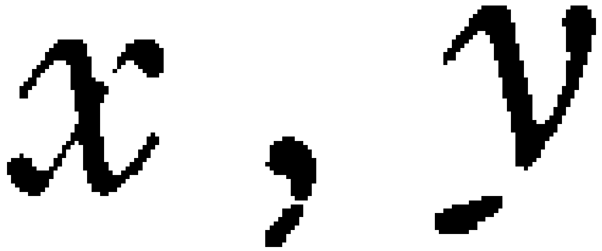


Figure 4.1: Examples of fractured symbols from example expression 2. Some symbol classes will have common points of fracturing, but each fracture is caused by random noise.

was the context only histogram size, and therefore amount of context viewed.

The second set of experiments for multi scale context showed higher accuracy rates in the cross validation. At this point the accuracy for edge merge/split classification ranged around 99.7% and the variations between parameters were less than a tenth of a percent. The context range chosen from this experiment was 1.75x the base radius for fitting the parent and child. In comparing the object recall rate when segmenting the testing set there was not much of an increase, less than a tenth of a percent. The change in precision was also small, but it did improve more than the recall, from 99.20% to 99.23%. I found using two shape context histograms worked best, giving a symbol recall rate of 99.47% and precision 99.23%. These results make sense when looking at the type of errors in segmentation.

4.3.2 Results Analysis

Segmentation of typeset symbols has two kinds of merges between connected components. There are valid merges between symbols with traditionally more than one connected component and there are merges between symbols that have been fractured into multiple connected components. The fracturing is a result of noise and can have high variation in how symbols are fractured. Symbol shape is stable, resulting in stable appearance of valid merges. The most frequent cases where valid merges fail are lower case i and double prime. In both cases the individual connected components of the symbols look like other symbol classes. Look at the lower case i errors in more detail show the missed merges occur when the i is a subscript. This puts the dot for the i at about the position you would expect to see a period, possibly creating some confusion in the contextual information.

Fractured symbols produce the majority of errors in segmentation of type set symbols. While some symbols have common weak points where fracturing is most likely to occur there is still a lot of variation in the fractures. This high variation might explain why lower resolution histograms did better and why shape context histograms did better for extended context. The shape context has higher resolution (smaller and closer bins) towards the center and lower resolution further away. Large differences in location for further out context or even parent and child components affect values less in the lower resolution portion of the histogram. Shape context histograms are also better at capturing direction information with the angle sectors. For grid histogram depending on the relative sizes of the histogram a corner might represent different angle for different histograms, preventing reliable comparisons for direction.

When the multi scale context is used the correct classifications increase, but only a small increase in recall of symbols is seen. Looking at the errors the explanation for this is that the fractured symbols not being properly merged are getting more pieces together, but still end up without all the parts together. What makes the fractured symbols segmentation problem hard is the high variance and low number of cases available to train on. Over the entire dataset each symbol might have relatively few fractured cases and each can be a little different from the others. overall segmentation was able to do well with relatively few, 122, simple feature.

4.4 Symbol Classification

The experiments for symbol classification looked to answer several questions. Is using grid histograms or shape context histograms better for symbol classification? Which provides better classification when using shape context histograms, point counting method or parzen method? What is the behavior of the histograms as resolution is increased? Does using context help symbol classification? Is there an amount of context that is detrimental to symbol classification?

The result showed the extremely regular shape of typeset symbols allows for high classification rates with either of the histogram types and with all of their methods for measuring densities. With a single visual density histogram, grid histogram, shape context with point counting, or shape context with parzen, a recognition rate of 98.2% is achievable with high resolution. When context was added the recognition rate went to 99.3% and neither the histogram for the symbol density or the context had to be as high resolution. Context allowed better recognition with less feature values. The results are inconclusive in regard to deciding on the best visual density feature to use, but some important behaviors can be seen in these experiments' results. Behaviors of the histograms for resolution and size changes and comparison of the two kinds of visual density histograms.

Experiments for symbol classification focus only on the which features to use and best parameters to use with those features. The experiments used the ground truth segmentation for all the symbols. Experiments used cross validation with the training set to compare the different feature sets using the metric of symbol classification rate. The symbol classification rate measures how many of the symbols being classified are given the correct classification. After cross validate finds the best parameters and features sets testing is done on the testing dataset.

4.4.1 Experiments

The first experiments used single histograms to capture the shape density of the symbol being classified. In the experiments the resolution and method of measuring values for the histogram were varied. grid histograms were test with resolutions from 2x2 to 11x11, using the fuzzy histogram method. Shape context histograms were tried using combinations of 1 to 5 circles and 2 to 18 angle sectors. The measuring methods used with the shape context histograms were point counting and Parzen. When Parzen was being used with the shape context features a narrow range of sigma factors were tested, from .05 to .15. This range of sigma factors was shown to be best for the high resolution shape context histograms.

The results from the first experiments show that for each type of histogram tried there is a similar behavior as resolution is increased and each was able to reach the same maximum symbol recognition rate at 98.2%. The universal pattern was as resolution increased the recognition rate increased, until leveling off around the maximum at the highest resolutions. At the lowest resolutions the grid histogram did much better than both shape context histograms. Also at the lower resolution the point counting shape context histogram did better than the Parzen, but this might be due to the selection of sigma factors. At the

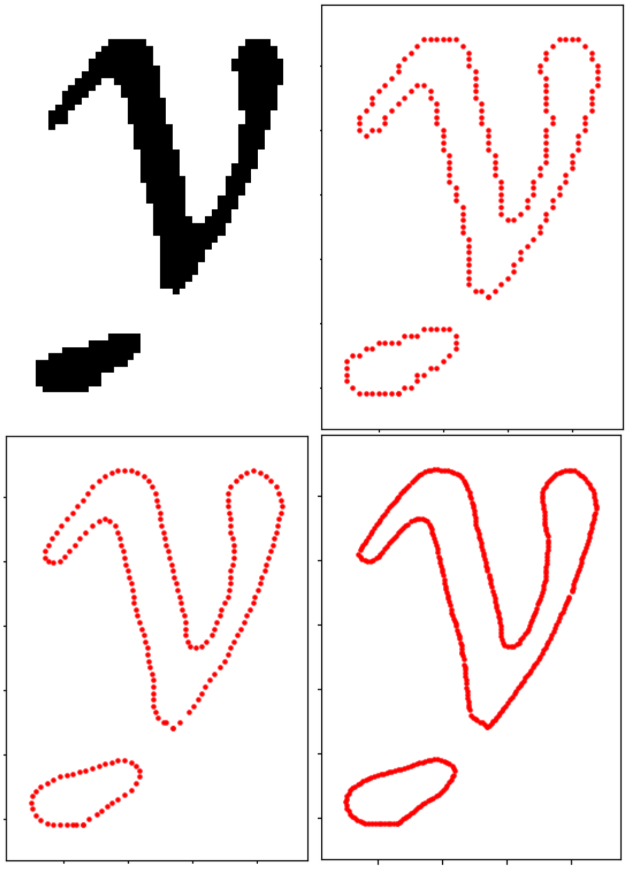


Figure 4.2: A processed fracture ’y’ symbol from example expression 2. (Top Left) The original symbol image. (Top Right) The extracted contour of the symbol image. (Bottom Left) The smoothed contour using smoothing distance of 2 to average point locations with points before and after in sequence. The smoothed contour has the same number of points. (Bottom Right) The symbol contours after using Catmull splines to do more smoothing and resample the contours.

medium resolutions, having 25 to 49 feature values, the grid histogram and Parzen shape context did the best with the grid histogram doing slightly better in one case. Though at these resolutions the difference in recognition rates between histograms are from .1% and .5%. After this point the recognition rates level off as resolution is increased and differences between the histograms drop to <.1%. Even at the lowest resolutions the worst histograms achieved >90% accuracy in symbol recognition, showing how effective these simple features are for typeset symbol recognition. The fact that at high resolutions even methods like point counting level off in accuracy instead of dropping seems to reflect the regularity of typeset symbols' shape.

The next set of experiments aimed at adding contextual information as a way to improve the recognition rate. The experiments also explored how much context is actually helpful and what effects resolution have. For the experiments two histograms would be used, one to capture the target symbols shape and anther to capture the contextual information from around the target symbol. The histogram for symbol shape was kept constant throughout these experiments as a 5x5 grid histogram using the fuzzy method of measuring values at the corners. In each of the experiments a shape context histogram was used for the contextual information. The contextual histogram varied its resolution and size for the experiments. The resolution varied from 1 circle and 2 angle bins to 5 circles and 16 angle bins. The size of the histogram ranged from 1 to 8 times the base radius needed to fit all the target symbol's points within a circles centered on the symbol.

With the second set of experiments the contextual information showed improved symbol classification, even when low resolution contextual information was used. On its own without the contextual histogram the 5x5 grid histogram achieves a symbol recognition rate of 97.4%. When contextual information is added using a shape context histogram with 2 circles and 6 angle sectors the symbol recognition rate increased to 98.8%. This shows that having contextual information along with the symbol visual density can do better than symbol visual density on its own and with less feature values. As the resolution of the contextual information increases the recognition rate also increases. Like the symbol visual densities increasing the resolution has diminishing returns, but no downward trends in recognition rate were seen. The difference in recognition rate between low resolution and high resolution contextual histogram was between .5% and 1.2% depending on its size. The other parameter which was varied for the contextual histogram was its size. As the size increased and more contextual information was included the recognition rate increased. The exception to this trend is very low resolution histograms did worse when size was increased. The difference in symbol recognition was <.5% between the smallest and largest contextual histograms. In combining the symbol density and contextual features symbol recognition reached 99.3%. This was done using a 8x8 grid histogram using fuzzy point measurements for the symbol's shape and a shape context features histogram using point counting with 5 circles, 14 angle sectors, and radius factor of 8 for context only.

4.4.2 Results Analysis

Classification based on symbol shape alone has the obvious problem of trying to distinguish symbol classes with the same or similar shape. Even with the very high resolution histograms for visual density there are many cases of this problem. Looking at the errors for the best symbol features only symbol recognizers the classes causing the most confusion are the ones with a simple horizontal line shape. The classes which all share this shape are

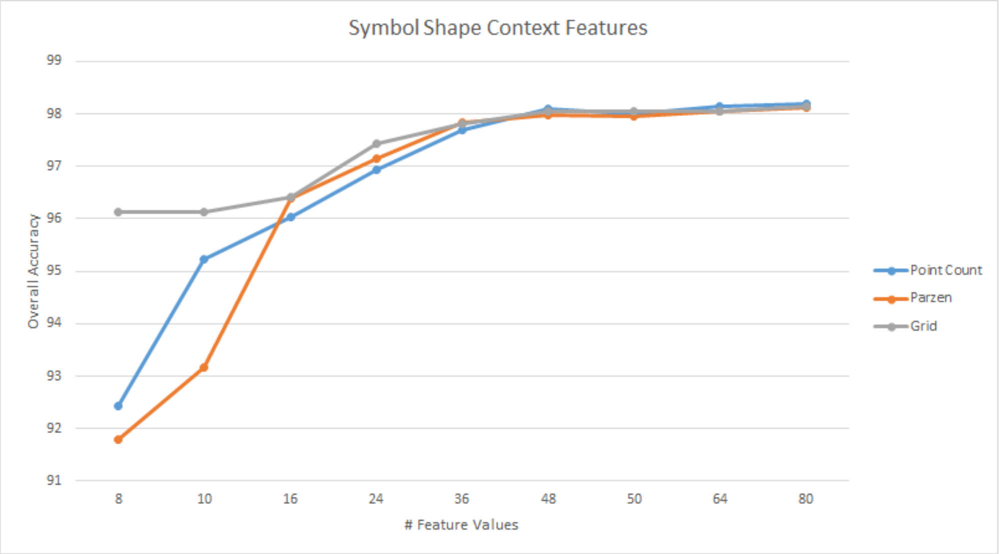


Figure 4.3: Graph comparing the symbol recognition rate when using different histograms for visual symbol density features. These histograms to capture symbol shape density are the only features used in symbol recognition. X axis is the symbol classification accuracy, the y axis is the number of feature values (aka. resolution) in the histogram and used in the classifier. 2D grid (Grey line) does best at low resolution. All three converge at high resolutions.

fractional line, minus, overline, and hyphen. Confusion between these classes makes up 40% to 50% of the errors. The next set of most confused symbol classes are dot, cdot, and period. These make up another 10% of errors. Other common confusions are letters with similar lower and upper cases, such as ’S’, ’P’, and ’O’. For the horizontal line classes there is little to no way of telling them apart by looking at the symbols in isolation.

Contextual information was able to cut down on the errors caused by classes with the same or similar shape. The confusion between the horizontal line classes was almost completely eliminated. There was still come confusion between cdot and period. When looking at the actual cases where these errors occur you can see that the confused symbols are relatively far from any other symbol. The contextual histogram size is based on the size of the symbol being classified, so with smaller symbols less context information is captured. Some of the remaining errors in symbol classification still include traditional difficult classifications, such as capital and lower case letters. For similar shapes there are a few distinct cases. Lower case 'L' is being mistaken for one, parentheses, and 'I', periods and commas have a few confusions. With such high classification rates more and more of the remaining errors are linked to printing and scanning noise in the dataset. A larger portion of errors are from fractured symbols. The fracturing of symbols that are usually one connected component into several was seen to be a problem in segmentation as well. In classification even if the fractured symbol is properly segmented it may not be classified correctly. Almost 25% of the remaining classification errors involve fractured symbols.

4.5 Expression Parsing

The experiments for parsing expressions looked at both adapting previously used approaches and features, as well as adding new ways of parsing. The initial experiments for parsing used the ground truth segmentation and symbol classification. First set of experiments looked at reproducing the approached used for handwritten expression parsing [11] and adapting the features to typeset expressions. The adapting of features includes introducing the grid histogram to expression parsing, as it was originally used for symbol recognition. The second set of experiments looked at how modified representations of expressions structures could help in parsing them. The next set of experiments looked at using multiple stages of parsing with a cascade of classifiers. Experiments were also done to look at using different forms of contextual information to do parsing, with a focus on baseline information. The final experiments looked at a simple form of adding unique symbol relation constant to the parsing to prevent results from having invalid expression structures.

The experiments for parsing either focused on the features, and their parameters, used for classifying relations or on the parsing algorithms itself. The experiments done on features consisted of first doing cross validation testing with a search over the space of parameter values for the features and then doing parsing tests with the best performing parameters from the cross validation. The experiments for the parsing algorithm only looked at parsing results. In the cross validation the metric used for evaluation is the classification rate of all relations in the validation set. This includes both the actual relations and false relations found when building the line of sight graph. The false relations have a ground truth relation of NoRelation. The parsing has a few metrics for evaluation. Two different kinds of correctness are looked at, detection and detection with classification. Detection refers to correctly identifying that two symbols have a relation between them in the valid layout tree. For detection the classification of the relation doesn't matter. It is in the second, detection and classification, where the relation between the to symbols is a valid relation and correctly classified. These two measure can be done at the relation level or expression level. At the relation level the rates refer to what proportion of all relations in all expression being tested are correct. For expression level the measure is the proportion of test expression with no errors for that measure.

4.5.1 Experiments

Adapted Parsing Approach. The original approach[11] to parsing used geometric features along with visual density histogram to classify relation edges in a line of sight graph.

This histogram used was a shape context histogram. The shape context histogram was used to capture visual density features for the parent symbols location, child symbols location, and surrounding context. In addition to these feature types a second kind of visual density histogram was added, the two dimensional grid histogram. The grid histogram could be used in place of the shape context histogram to get the parent, child, and context visual density information.

Comparison of the different features was done by using each individually for the classification of relations between symbols. The original set of geometric features consisted of 38 values. The set of geometric features was then expanded to 50 values. In the cross validation testing it was shown that the expanded set of geometric values did better in classifying the relations between symbols.

The density histograms were tested independently with different resolutions and sizes. In the testing the histogram would collect all three set of values, parent, child, and context. The grid histogram resolutions tested in cross validation ranged from 6x6 to 10x10 and sizes from 1.0 to 1.5 times the base size. The shape context histogram resolution ranged from 4 circles and 6 angle sectors to 6 circles and 12 angle sectors. The sizes tried for the shape context went from 1.0 to 2.0. Both histograms used the Parzen method for measuring density values. In each case increasing the resolution helped. Also using more than the base size of the histogram helped in classifying relations when using shape context but not grid histograms. Increasing the grid sizes lowered classification. For grid histograms 1.00 times the base size worked best and for shape context 1.5 times worked best. What was not tried was a smaller histogram for just the parent and child density features and a larger one for context. The grid histogram seemed to have an advantage when using smaller histograms, but as size increased the shape context improved more. Both histograms did better than the geometric features on their own. Looking at expression level detection and classification the geometric features got 90.54%, shape context features 93.25%, and grid histogram 93.46%.

When looking at just the geometric features the cross validation indicated that including the added features helped in relation classification. Looking at results on the testing set agree, with a slight increase in relation classification from 97.25% to 97.56%. A greater increase was seen in the expression rate when the new geometric features were used, 90.54% from 80.52%. The geometric features were out performed by both the shape context histograms and grid histograms in the cross validation.

Experiments looking using only a density histogram for parsing saw grid histograms perform better than shape context with smaller, base sized histograms, but worse for larger histograms. In the cross validation relation classification ranged from 98.9% to 99.22%. When the size of the histograms was kept at the base size, just big enough to fit both the parent and child symbol, The grid histogram seemed to do slightly better than shape context, by about .1%. In testing the expression rate rose from 93.24% to 93.46% when a 7x7 grid histogram was used instead of a shape context with 6 circles and 12 angle sectors. As the size of the histograms increased the gird histogram did worse than before and the shape context did better in cross validation. There are a few factors that might account for grid histograms performance in larger histograms with more context. The variance in positioning of context might increase further out from the center and cause a grid with uniform bin sizes problems. Shape context have larger bins further from the center and this might help with the higher variance for the context. Also as grids grow larger all the bins grow larger at the same rate, meaning less detail every where. This would include the parent and child spatial density features. In shape context the outer bins are affected more than the inner bins terms of feature details.

After examining the features in isolation they were combined to use both the extended geometric features along with a visual density histogram. When using a grid histogram the base size was used, but the shape context used a size of 1.5 times the base size. The resolution for the grid was 10x10 and the shape context used 6 circles and 12 angle sectors. For both sets of features the experiments showed about the same recognition rates. The grid histogram covers more area than the shape context of the same base size, so at the tested sizes the shape context actually has more contextual information. It may be that the grid histogram does better for the parent and child spatial densities, but worse for context because of higher variance in the context. No experiments were done to exclude the context information when using the grid histogram and use a shape context histograms for all contextual information.

Alternate Expression Representations. These experiments were done to test the alternate ways of representing the structures of expressions. The two changes made to the structure were removing baseline punctuation and accents from the main baseline. Baseline line punctuation symbols, commas, periods, and ldots, from the main baseline to their own baseline with their own relation class. This means instead of having a comma horizontal the the symbols it is separating it would be considered like a subscript and the two symbols would now be horizontal to each other. Accents instead of being horizontal to the symbol before them would be considered above the symbol under them, then the symbol under them would be horizontal to the previous symbol. (see structure changes figure). The experiments were run using geometric features combined with a grid histogram for visual density features. Using these two changes to the expression representation increased relation detection, classification, and expression structure recognition. The best expression level detection and classification rates when up by 2% when going from expression v1b to v5, see Table 4.3.

Alternate expression representations experiments show when the expression structure were changed to better represent the spatial relations with baseline punctuation and accents the relation classification and expression recognition increased. The feature set used included the extended geometric features and a grid histogram using parzen with resolution 10x10, at the base size. The grid histogram captured the parent, child, and context visual densities. The original expression, see v1b in Section 4.1, detected 98.86% of the correct relations, had 98.51% of relations correctly classified, and 93.65% expression recognition rate. With the modified expressions, version 5, relation detection went to 99.29%, relation classification went to 98.98%, and the expression rate was 95.31%. In number of errors regarding baseline punctuation and accents dropped with the new expression structures, with the number of relation errors in the parsed expressions decreasing by 33%.

Looking at the change in error cases for the baseline punctuation there are more relations detected and correctly classified than before. The two main errors with punctuation were adding it to the end of a subscript baseline or incorrectly classifying the relation as RSUB instead. The first error occurred when there was a subscript symbol between the parent symbol and the baseline punctuation, which makes it look like the punctuation is horizontal to the subscript symbol. If there was no subscript symbol in the way then the relation looks more like a subscript relation visually. With the change to the structure by having the punctuation have its own relation, PUNC, both cases were reduced. there are still cases of both, but now when the PUNC relation is misclassified as RSUB it can be fixed in post processing by using the classification of the symbols to correct all relations to baseline punctuation to be PUNC. There are also a few classes of non-baseline punctuation symbols with a relation of PUNC which should be RSUB. These can also be fixed with the same kind of post processing.

The changing of the relations dealing with accents has many different effects. The first was an increase in relation recall for the line of sight graph. There were cases where the symbol under the accent would block line of sight to the parent symbol or child symbols of the accent. One of the major errors with accents was the parent symbol choosing a horizontal relation to the symbol under the accent instead of the accent itself. The change in structure made the ground truth match the horizontal sequence of symbols on writing lines. In the confusion histogram the number of incorrectly classified false relations (NoRelation) as horizontal dropped by 40%. The number of Horizontal relations miss-classified as NoRelation dropped by 50%. The other relations which improved are the child relations of the accent, which were changed to have the symbol under the accent as the parent. Previously any RSUB or RSUP relation belonged to the accent, but would be classified as belonging to the symbol under the accent. All of these errors were fixed by the change in expression structure. The few remaining error cases for accents happen when there is multiple symbols under the accent and incorrect one is chosen to have the upper relation from the accent to it. Again this is something that can be fixed in post processing, but currently there are only 6 cases of this in the test set.

Before the structural changes the accent errors made up the majority of the top errors, closely followed by baseline punctuation. After the structural change there are still some baseline punctuation errors, but the top errors belong to line of sight errors caused by blocking subscripts and superscripts. These are relations that are not even detected for classification when the line of sight graph is built. Another frequent problem is subscript symbols being given horizontal relations, some of these cases overlap with the block subscripts and others are just subscripts which seem to be on the same vertical level as their parent symbol. More than 50% of the remaining errors are either horizontal relations not being detected or false relations being classified as horizontal.

Cascade Classifiers. The experiments used multiple classifiers to do a sequence of classifications of relations before extracting the final expression structure. In the experiments two classifiers were used. The first classifier is used to try and identify relations as a horizontal or not horizontal. The second classifier is used on all relations classified by the first as not horizontal. The idea is the binary classification of horizontal relations would be easier. Both classifiers use geometric features and shape context features. The experiments showed no improvement in relation classification, indicating this simple approach to a cascade wouldn't help. This approach might be improved by adding unique symbol relation constraint which restrict the relations chosen and eliminate invalid ones for the next round of classification. Another approach would be instead of a cascade use a set of binary classifiers for all the relations.

With most of the relations in expressions being horizontal and most errors involving the horizontal relation it is the relation where the most overall improvement can be gained. Unfortunately using the cascading approach didn't show any increase in relation classification or expression recognition. More horizontal relations were correctly classified, but at the same time there were more miss classifications of horizontal. Both other real relations between symbols and false relations between symbols that should be excluded from the final expression structure were classified as being horizontal relations. This might be an effect of the high prior for horizontal relations, 78% of all relations. It might be possible to get improvement from the cascading classifier if unique symbol relation constraint was used to restrict the selection of horizontal relations. currently any number of horizontal relations can be selected for a symbol in the first pass of relation classification.

Symbol Shape Context. These experiments aimed to use the parent and child's shape visual density as additional context in classifying the relation between them. These features were used along side the geometric and a grid histogram for the parent, child, and context information.

The symbol visual densities for the parent and child symbols were used as additional context for the classification of relations between two symbols. The visual densities used for the individual parent and child symbols were the same as those used in symbol recognition and captured the shape densities of the symbol. The experiments combined either the child or parent shape densities with the geometric features and grid histogram for capturing the spatial density features for the parent, child, and context. Grid histograms were used the parent and child shape density features. The resolutions were varied in the experiments from 5x5 to 7x7. The experiments showed increased relation recognition and expression parsing when including the child shape visual density, but not when using the parent shape density.

symbol shape features as context for parsing turn out to be helpful for the punctuation relation classification. Relation classification and expression recognition was improved when a grid histogram of just the child symbol was added to the feature set. The cross validation however showed no improvement for the same features with the parent symbol. The cross validation indicated a lower resolution grid for the child was more helpful. The increases seen in the cross validation were less than .1%. When looking at parsing the test set expressions (version 5) when using the child shape density features a 6x6 grid is used, which adds to the geometric features and 10x10 grid histogram for spatial density. The testing results show the recall on correctly classified relations goes up by .1% to 99.08%. The expression rate goes up by .55% to 95.86%. Looking at the difference in errors between using the child shape and not it becomes apparent why it helped. The error cases where the child shape helped was baseline punctuation. In the version 5 expression the baseline punctuation has its own relation, but this would probably also help when punctuation had horizontal relations with its parent. The number of confusions between RSUB and PUNC relations go from 42 to 12. Many of the remaining errors for the PUNC relation are undetected relations that are completely missing from the final expression graphs. In many cases these missed relations are replaced by having the baseline punctuation horizontal to a subscript instead. A solution to this might be once again changing the expression structure to have baseline punctuation included as extensions of subscripts.

Unique Symbol Relation Constraint. The following experiments were done to see if applying unique symbol relation constraint when parsing could help resolve cases where invalid expression structures were being found when parsing. The most common error in these invalid expression structures was a parent symbol having multiple horizontal relations to child symbols, see Figure 3.14. The unique symbol relation constraint limits a parent symbol to only having at most one of each relation with it set of child symbols. If after classifying the relations in the line of sight graph there are more than one child with the same relation to the parent the one child with the highest probability from the classifier for the relation is kept. All other child symbols with the same relation are changed to their next most probable relation class. After all conflicts are resolved Edmonds' algorithm is used to find the maximum spanning tree as the final expression structure. The experiment was conducted using a classifier trained with the geometric features, grid histogram for spatial densities, and a grid histogram for the child symbol shape density. The experiment showed slightly improved relation classification and expression parsing.

Adding the unique symbol relation constraint to force symbols to not have duplicate relations to child symbols does help in parsing. The trade off in using this restraint is there are cases where when deciding between multiple child symbols with the same relation the wrong one is selected to keep that relation class and have the other child symbols change theirs the wrong one is selected. This will cause slightly lower relation detection and precision goes down for both relation detection and relation classification. Looking at the confusion histograms there are actually more errors than without the unique symbol relation constraint, 20% more errors. Despite this the expression rate actually goes up. The reason for this is that the expressions with harder error cases get worse, but the easier expressions are improved. When using the geometric features, 10x10 grid histogram for spatial densities, and the child symbol shape densities with a 6x6 grid histogram the expression recognition rate goes from 95.86% to 95.97%. The errors which are helped by the unique symbol relation constraint are subscripts being mistaken for horizontal relations.

4.6 Full Expression Recognition

Once each of the tasks for expression recognition had been tested individually, tests were done which did all three tasks. The segmentation would be done first on the primitive level symbol graph. Segmentation would merge the connected component nodes into symbol nodes. These symbol nodes would be used to construct a new line of sight graph. The symbol recognizer labeled each node. Last the parser would be used to get the final expression structure.

The same classifier and method was used to do the segmentation for all the tests. The segmenter used the 38 geometric features, a shape context features histogram with 3 circles, 6 angle sectors, and radius factor of 1.0 to capture the spatial density for the parent, child, and context, and a second shape context features histogram with 3 circles, 10 angle sectors, and radius factor 1.75 to capture more contextual information. The segmenter had a symbol recall rate of 99.47%.

The symbol recognition was kept the same for all tests as well. The features used were a 8x8 Grid histogram for the symbol density features and a shape context features histogram for context features with 5 circles, 14 angle sectors, and a radius factor of 8. On perfectly segmented symbols this classifier achieved 99.3% accuracy in symbol recognition. On symbol segmented by the segmenter outlined above 98.73% of all symbols were correctly classified.

A number of different parsers were tried for the full recognition. The results are outlined in the table below. The evaluation metrics being used are relation detection, relations classification, expression relation detection rate, expression relation classification rate, and expression structure rate. The relation detection indicates what percent of all relations in all the ground truth expressions were included in the parsed expressions. This does not include having the correct class for the relation, just whether two symbols which are suppose to have a relation between do. Relation classification is the percent of all relations that were correctly detected and classified. Expression detection rate indicates what percent of expression have no errors in detecting the correct relations between symbols. Expression classification rate indicates what percent of expressions have no relation errors, either detection or classification. Expression structure rate is the percent of expressions with no errors, segmentation, symbol recognition, or parsing.

4.7 Summary of Experiment Results

The experiments showed the effectiveness of Hierarchical Contextual Parsing for recognition of scanned typeset expressions using maximum spanning tree extraction with simple classifiers. The contextual features used for each of the recognition tasks helped in their classifications and in it appeared the shape context histograms were better able to handle the variance of context than the grid histograms, see Subsections 4.3.2 and 4.5.1. Segmentation seems to benefit from multiple scales of context, one right around the parent and child connected components and a second which extends further out into the surrounding area with a radius 1.75 times larger than the first set of context. Parsing benefited from additional context is a different way. When adding a visual density histogram to capture the child symbols shape it improved classification of PUNC relations, see symbol shape context in subsection 4.5.1. For symbol classification a large amount of context, radius 8 times the symbol radius, can be used with shape context histograms. Beyond the features themselves additional factors were found to help in parsing the typeset expressions. One such factor was the layout representation of the expressions. Changing how baseline punctuation and accent relations are represented, see Section 3.1, also improved recognition. The other addition to parsing was using a unique symbol relation constraint, see Section 3.6, to remove invalid expression hypotheses increased expression recognition rates. The final expression recognition rate starting from connected component level was 90.83%.

Table 4.3: Detection(Det) and Detection with Classification(Det+Class) results for parsing expressions using geometric and density histogram features. Density histograms - shape context features:scf(circles,angle sectors,radius factor), Grid histogram:2dh(rows x cols, size factor). Ground truth representation(GT Rep) refers to the verson of expresstion version used as ground truth. v1: Original Expressions, v1b: Trailing Punctuation has been removed, v5: Trailing Punctuation has been removed and Baseline and Accent restructuring has been done.

Parsing TypeSet Expressions					
		Relations		Expressions	
GT Rep	Features	Det	Det+Class	Det	Det+Class
	Parsing with Given Symbols				
v1b	38 Geometric,SCF(5,6,1.5)*	98.44	98.09	93.06	91.96
v1b	38 Geometric,SCF(5,6,1.5)	98.74	98.34	94.33	93.16
v1b	50 Geometric,2DH(10x10,1.0)	98.86	98.51	94.8	93.65
v1b	50 Geometric,2DH(10x10,1.0),Child-2DH(6x6)	98.85	98.58	94.6	93.88
v1	50 Geometric,2DH(10x10,1.0),Child-2DH(6x6)	98.81	98.55	94.35	93.66
v5	50 Geometric,2DH(10x10,1.0)	99.29	98.98	96.5	95.31
v5	Geometric,2DH(10x10,1.0),Child-2DH(6x6)	99.32	99.09	96.66	95.86
v5	Geometric,2DH(10x10,1.0),Child-2DH(6x6)**	99.3	99.07	96.54	**95.97**
	Parsing with Segmented Symbols				
v1b	38 Geometric,SCF(5,6,1.5)*	97.56	97.25	91.19	90.23
v1b	38 Geometric,SCF(5,6,1.5)	97.87	97.54	92.53	91.48
v1b	50 Geometric,2DH(10x10,1.0)	97.95	97.64	92.78	91.77
v1b	50 Geometric,2DH(10x10,1.0),Child-2DH(6x6)	97.95	97.73	92.69	92.05
v1	50 Geometric,2DH(10x10,1.0),Child-2DH(6x6)	97.82	97.6	92.15	91.68
v5	50 Geometric,2DH(10x10,1.0)	98.36	98.08	94.35	93.38
v5	Geometric,2DH(10x10,1.0),Child 2DH(6x6)	98.4	98.19	94.52	93.89
v5	Geometric,2DH(10x10,1.0),Child 2DH(6x6)**	98.37	98.18	94.41	**93.95**

* The line of sight graph does not use extra search to find baseline punctuation

** The unique symbol relation constraint is used during parsing

Table 4.4: Expression Rates for Full Recognition of Typeset Math Expressions using geometric and density histograms. Density histograms - shape context features:scf(circles,angle sectors,radius factor), Grid histogram:2dh(rows x cols, size factor). Ground truth representation(GT Rep) refers to the verson of expresstion version used as ground truth. v1: Original Expressions, v1b: Trailing Punctuation has been removed, v5: Trailing Punctuation has been removed and Baseline and Accent restructuring has been done.

Full TypeSet Expression Recognition		
Expr GT	Parsing Features	Expr Rate
v1b	38 Geometric,SCF(5,6,1.5)*	87.28
v1b	38 Geometric,SCF(5,6,1.5)	88.46
v1b	50 Geometric, 2DH(10x10,1.0)	88.81
v1b	50 Geometric, 2DH(10x10,1.0),Child-2DH(6x6)	89.02
v1	50 Geometric, 2DH(10x10,1.0),Child-2DH(6x6)	88.45
v5	50 Geometric,2DH(10x10,1.0)	90.31
v5	Geometric,2DH(10x10,1.0),Child-2DH(6x6)	90.76
v5	Geometric,2DH(10x10,1.0),Child-2DH(6x6)**	**90.83**

* The line of sight graph does not use extra search to find baseline punctuation

** The unique symbol relation constraint is used during parsing

Chapter 5

Conclusion and Future Work

Our results provide strong support for typeset math expression recognition being done effectively by maximum spanning tree extraction with simple classifiers using spatial and visual density features. These visual density features have been seen as very effective for the segmentation and recognition of symbols, along with symbol relationships. The visual approach to these tasks is aided by restructuring the expressions to better match the actual visual layout. These structural changes include moving baseline punctuation to their own baseline with a new relation and changing accents to only having an above relation with the symbol under them. One of the major difficulties in using a line of sight graph with typeset expressions is the crowded baselines, often caused by more horizontally adjacent subscripts. Modifications to the line of sight algorithm can improve the relation recall and have been shown to improve expression recognition rates even with loss to precision, resulting in more invalid edges in the set of hypotheses.

Additionally exploring forms of contextual information shows promising improvements to typeset expression recognition. The context used in symbol recognition has shown the ability to improve recognition beyond what only shape features achieved. Having two scales of context for segmentation also improved symbol recall over using less context. In parsing the attempts at using baseline specific context with maximum spanning tree context and directonally extended context showed more confusions involving the dominant relation, horizontal. It was found that using a visual density histogram to capture the shape of the parent or child histogram can help with parsing relations that have a high correlation to certain symbol classes. In the case of the relation class specific for baseline punctuation looking at the child symbol's shape helped reduce false positives and false negatives for the relation when parsing. From benefits seen in using the context information from shape context features and grid histograms, visual context still seems to be a promising area to look for features to improve approaches to expression recognition.

Looking beyond discriminative features to improve relation classification, a common approach to parsing is using language constraints. The implemented language constraint of have only one of each relation type between a parent and its child symbols prevented invalid expressions where a parent could have the same relation to multiple child symbols. Using this constraint was able to improve expression recognition rates in testing. Including more knowledge about expressions and their structures seems to be a good way to supplement the approach presented above.

5.1 Future Work

Considering what has been seen and learned from this work there are a number of possible future works that would seem promising. The current system does all the recognition tasks in one pass and further more the segmentation and parsing are done without any knowledge of symbol class. They don't use symbol classification information or scores from the classifier used for symbol recognition. It is common for this information to be utilized in the segmentation and parsing. With the observed levels of symbol recognition in our system it should be examined if this information could be used to improve the segmentation or parsing. The symbol classes or class confidences could be used directly, in language constraints, or for syntax-based features. To an even greater extent information can be shared from all three tasks to help recognition. One way of doing this is have multiple iterations of recognition, where after each round the results are saved in features for the next round. This would be an extended Hierarchical Contextual Parsing(HCP) approach.

A major part of the hierarchical contextual parsing approach is the visual density features used for all three recognition tasks. The changes made to expression structures to better take advantage of them seems like a step in the right direction. There are probably additional changes or alterations that could be made to the layout structure of expressions. One such change is to have the punctuation fully merge with the subscript baselines. A common error is to see the punctuation as a horizontal extension to the end of baselines. Changing the expressions to reflect this could see further improvements and would eliminate the need for the special line of sight rule to look for blocked punctuation. A slightly different way of looking at modifications to the visual density would be masking the visual space to something other than euclidean space. Visual features could then be taken from this space as an alternative or in addition to the features in the original space.

One of the things shown is the effectiveness of simpler classifiers, a random forest in our case, could be used in the classifications. Moving forward more complex classifiers could be used with the same set of features or just layout structure. One candidate would be using neural networks trained on the spatial and visual features to try and improve on the difficult and edge cases, like subscripts located in the horizontal baseline, in the expression recognition. Another purpose of deep learning with networks could be learning new visual density features from a convolutional neural network. Moving from the space of understanding the problem and how it can be approached with simple methods allows for a lot of possible ways forward.

Appendix

Appendix 1

Grid Search Appendix

1.1 Grid Searches for Segmentation

Grid search properties:

5 fold cross validation over training set.

Classifier: Random Forest (50 Trees, 40 Max Depth, 30 Max Features Selected, Gini Split Criteria)

Grid Search 1:

Lei Hu's 38 Geometric Features

Two Dimensional Grid Histogram (Parent, Child, Context)

Parzen with Sigma Factor of .1

Resolutions : [5x5, 6x6, 7x7, 8x8, 9x9]

Sizes: [1.0, 1.25]

Grid Search 2:

Lei's 38 Geometric Features

Two Dimensional Grid Histogram (Parent, Child, Context)

Parzen with Standard Deviation Factor of .1

Resolution: 5x5

Size: 1.0

Shape Context Features (context only)

Parzen with Sigma Factor of .1

Number of Circles: [2, 3, 4]

Number of Angle Sectors: [6, 8, 10]

Radius Factor: [1.25, 1.5, 1.75, 2.0]

Grid Search 3:

Lei Hu's 38 Geometric Features

Two Dimensional Grid Histogram (Parent, Child, Context)

Parzen with Standard Deviation Factor of .1

Resolution: 5x5

Size: 1.0

Two Dimensional Grid Histogram (Child)

Parzen with Standard Deviation Factor of .1

Resolution: [5x5, 6x6, 7x7]

Grid Search 4:

Lei Hu's 38 Geometric Features

Shape Context Features (Parent, Child, Context)

Parzen with Sigma Factor of .1

Number of Circles: [3, 4, 5]

Number of Angle Sectors: [6, 10]

Radius Factor: [1.0, 1.5]

Grid Search 5:

Lei Hu's 38 Geometric Features

Shape Context Features

Parzen with Sigma Factor of .1

Number of Circles: 3

Number of Angle Sectors: 6

Radius Factor: 1.0

Recorded Values: [(Parent and Child), (Parent, Child, Context)]

Shape Context Features

Parzen with Sigma Factor of .1

Number of Circles: [3]

Number of Angle Sectors: [10]

Radius Factor: [1.25, 1.5, 1.75]

Recorded Values: [(Context Only), (Parent, Child, Context)]

1.2 Grid Searches for Classification

Grid search properties:

5 fold cross validation over training set.

Classifier: Random Forest (50 Trees, 40 Max Depth, 30 Max Features Selected, Gini Split

Criteria)

Grid Search 1:

Shape Context Features (symbol)

Parzen with Sigma Factor: [.075, .1, .125]

Number of Circles: [1, 2, 3, 4, 5]

Number of Angle Sectors: [2, 4, 6, 8, 10, 12, 14, 16]

Grid Search 2:

Shape Context Features (symbol)

Point Counting

Number of Circles: [1, 2, 3, 4, 5]

Number of Angle Sectors: [2, 4, 6, 8, 10, 12, 14, 16]

Grid Search 3: Grid Histogram (symbol)

Size: [2x2, 3x3, 4x4, 5x5, 6x6, 7x7, 8x8, 9x9, 10x10, 11x11]

Grid Search 4:

Grid Histogram (symbol)

Size: 5x5

Shape Context Features (context)

Parzen with Sigma Factor: [.1]

Number of Circles: [1, 2, 3, 4, 5]

Number of Angle Sectors: [2, 4, 6, 8, 10, 12, 14]

Radius Factor: [1.0, 2.0, 4.0, 8.0]

Grid Search 5:

Grid Histogram (symbol)

Size: 5x5

Shape Context Features (context)

Point Counting

Number of Circles: [1, 2, 3, 4, 5]

Number of Angle Sectors: [2, 4, 6, 8, 10, 12, 14]

Radius Factor: [1.0, 2.0, 4.0, 8.0]

1.3 Grid Searches for Parsing

Grid search properties:

5 fold cross validation over training set.

Classifier: Random Forest (50 Trees, 40 Max Depth, 30 Max Features Selected, Gini Split Criteria)

Grid Search 1:

Geometric Features: [38 Features, 50 Features]

Grid Search 2:

Shape Context Features (Parent, Child, Context)

Parzen with Sigma Factor of .1

Number of Circles: [4, 5, 6]

Number of Angle Sectors: [6, 8, 10, 12]

Radius Factor: [1.0, 1.5, 2.0]

Grid Search 3:

Two Dimensional Grid Histogram (Parent, Child, Context)

Parzen with Standard Deviation Factor of .1

Resolution: [6x6, 7x7, 8x8, 9x9, 10x10]

Size: [1.0, 1.25, 1.5]

Grid Search 4:

Geometric Features: [38 Features, 50 Features]

Shape Context Features (Parent, Child, Context)

Parzen with Sigma Factor of .1

Number of Circles: [5, 6]

Number of Angle Sectors: [10, 12]

Radius Factor: [1.0, 1.5, 2.0]

Grid Search 5:

Geometric Features: [38 Features, 50 Features]

Two Dimensional Grid Histogram (Parent, Child, Context)

Parzen with Standard Deviation Factor of [.08, .09, .1, .11]

Resolution: [7x7, 8x8, 9x9, 10x10]

Size: [1.0]

Grid Search 6:

50 Geometric Features

Two Dimensional Grid Histogram (Parent, Child, Context)

Parzen with Standard Deviation Factor of .1

Resolution: 10x10

Size: 1.0

Two Dimensional Grid Histogram (Child)

Parzen with Standard Deviation Factor of .1

Resolution: [5x5, 6x6, 7x7]

Grid Search 7:

50 Geometric Features

Two Dimensional Grid Histogram (Parent, Child, Context)

Parzen with Standard Deviation Factor of .1

Resolution: 10x10

Size: 1.0

Two Dimensional Grid Histogram (Parent)

Parzen with Standard Deviation Factor of .1

Resolution: [5x5, 6x6, 7x7]

Grid Search 8:

50 Geometric Features

Two Dimensional Grid Histogram (Parent, Child, Context)

Parzen with Standard Deviation Factor of .1

Resolution: 10x10

Size: 1.0

Shape Context Features (context only)

Parzen with Sigma Factor of .1

Number of Circles: [6]

Number of Angle Sectors: [12]

Radius Factor: [1, 1.25, 1.5, 1.75, 2.0, 2.25]

Bibliography

[1] Francisco Álvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. Classification of on-line mathematical symbols with hybrid features and recurrent neural networks. In *Document analysis and recognition (icdar), 2013 12th international conference on*, pages 1012–1016. IEEE, 2013.

[2] Francisco Álvaro, Joan-Andreu Sánchez, and José-Miguel Benedí. Recognition of on-line handwritten mathematical expressions using 2d stochastic context-free grammars and hidden markov models. *Pattern Recognition Letters*, 35:58–67, 2014.

[3] Francisco Álvaro and Richard Zanibbi. A shape-based layout descriptor for classifying spatial relationships in handwritten math. In *Proceedings of the 2013 ACM symposium on Document engineering*, pages 123–126. ACM, 2013.

[4] Robert H Anderson. Syntax-directed recognition of hand-printed two-dimensional mathematics. In *Symposium on Interactive Systems for Experimental Applied Mathematics: Proceedings of the Association for Computing Machinery Inc. Symposium*, pages 436–459. ACM, 1967.

[5] R. G. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):690–706, Jul 1996.

[6] Kenny Davila, Stephanie Ludi, and Richard Zanibbi. Using off-line features and synthetic data for on-line handwritten math symbol recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pages 323–328. IEEE, 2014.

[7] Yuntian Deng, Anssi Kanervisto, and Alexander M Rush. What you get is what you see: A visual markup decompiler. *arXiv preprint arXiv:1609.04938*, 2016.

[8] Jack Edmonds. Optimum branchings. *Journal of Research of the National Bureau of Standards*, 71:233–240, 1967.

[9] Yuko Eto and Masakazu Suzuki. Mathematical formula recognition using virtual link network. In *Document Analysis and Recognition, 2001. Proceedings. Sixth International Conference on*, pages 762–767. IEEE, 2001.

[10] Ann Grbavec and Dorothea Blostein. Mathematics recognition using graph rewriting. In *Document Analysis and Recognition, 1995., Proceedings of the Third International Conference on*, volume 1, pages 417–421. IEEE, 1995.

[11] Lei Hu. Features and algorithms for visual parsing of handwritten mathematical expressions. *RIT*, 2016.

[12] Lei Hu, Kevin Hart, Richard Pospesel, and Richard Zanibbi. Baseline extraction-driven parsing of handwritten mathematical expressions. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 326–330. IEEE, 2012.

[13] Lei Hu and Richard Zanibbi. Line-of-sight stroke graphs and parzen shape context features for handwritten math formula representation and symbol segmentation. *Proc. ICFHR*, 2016.

[14] Lei Hu and Richard Zanibbi. Mst-based visual parsing of online handwritten mathematical expressions. *Proc. ICFHR*, 2016.

[15] Chinmay Jain, Lakshmi Ravi, Michael Condon, and Richard Zanibbi. Simple context features for handwritten and typeset math symbol classification are beneficial. *(Pending Review) International Journal on Document Analysis and Recognition (ICDAR)*, 2017.

[16] Harold Mouchere, Christian Viard-Gaudin, Richard Zanibbi, and Utpal Garain. Icfhr 2014 competition on recognition of on-line handwritten mathematical expressions (crohme 2014). In *Frontiers in handwriting recognition (icfhr), 2014 14th international conference on*, pages 791–796. IEEE, 2014.

[17] Harold Mouchère, Richard Zanibbi, Utpal Garain, and Christian Viard-Gaudin. Advancing the state of the art for handwritten math recognition: The crohme competitions, 2011—2014. *Int. J. Doc. Anal. Recognit.*, 19(2):173–189, jun 2016.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[19] R. Plamondon and S. N. Srihari. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*,

22(1):63–84, Jan 2000.

[20] Masakazu Suzuki, Seiichi Uchida, and Akihiro Nomura. A ground-truthed mathematical character and symbol image database. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 675–679. IEEE, 2005.

[21] Richard Zanibbi and Dorothea Blostein. Recognition and retrieval of mathematical expressions. *International Journal on Document Analysis and Recognition (IJDAR)*, 15(4):331–357, 2012.

[22] Richard Zanibbi, Dorothea Blostein, and James R. Cordy. Recognizing mathematical expressions using tree transformation. *IEEE Transactions on pattern analysis and machine intelligence*, 24(11):1455–1467, 2002.

[23] Richard Zanibbi, Harold Mouchere, and Christian Viard-Gaudin. Evaluating structural pattern recognition for handwritten math via primitive label graphs. In *Document Recognition and Retrieval*, pages 810–817, 2013.

[24] Ling Zhang, Dorothea Blostein, and Richard Zanibbi. Using fuzzy logic to analyze superscript and subscript relations in handwritten mathematical expressions. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on*, pages 972–976. IEEE, 2005.

[25] Siyu Zhu, Lei Hu, and Richard Zanibbi. Rotation-robust math symbol recognition and retrieval using outer contours and image subsampling. In *DRR*, 2013.