Context Sensitive Mathematical Character Recognition

Elena Smirnova *Stephen M. Watt*

Ontario Research Centre for Computer Algebra
The University of Western Ontario
London Ontario, Canada
{elena,watt}@orcca.on.ca

Abstract

This paper describes methods to increase the accuracy of mathematical handwriting analysis by using context information. Our approach is based on the assumption that likely expression continuations can be derived from a database of mathematical expressions and then can be used to rank the candidates of isolated symbol recognition. We present how predicted continuations for an expressions are derived, how they are combined with the recognition candidates, and the effectiveness of the results. We first review the techniques we have used to build and represent a mathematical context database. We then describe different strategies for combining context information with results obtained from the recognition of individual characters. Finally we present a summary of a case study, using a fixed dataset of common mathematical expressions to test the accuracy of on-line analysis.

1 Introduction

1.1 Motivation

To build a mathematical recognizer one would naturally wish to adopt the already well-developed techniques for natural language recognition. However, three major aspects of mathematical notation distinguish handwriting analysis of mathematics from that of natural languages. Most notably, general mathematical expressions occupy a two-dimensional layout, which implies a combination of drawing and writing techniques within a single formula. Secondly, much larger alphabets are used, wherein some characters are only slight variations of others. This makes the usual methods for text-based character recognition insufficient. Thirdly, there is no fixed mathematical vocabulary that may be used for disambiguation.

Dictionary-based methods are essential for recognition in natural languages: most text recognizers choose candidates for individual letters by matching possible combinations of candidates to dictionaries entries. This explains, for example, why the Microsoft text recognizer identifies the word in Figure 1 as "cloud", even though the first and last strokes are identical and can be equally as well recognized as "*cl*" or "*d*".

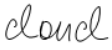


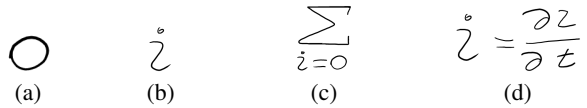
Figure 1. Ambiguity resolvable by use of dictionaries

1.2 Objectives

Just as character recognition in natural language text is improved by considering whole words, we wish to improve handwriting analysis for mathematics by using larger context. Although there is no fixed vocabulary of words, human readers will use domain knowledge of common expressions in their recognition. Our goal is to do the same thing automatically for mathematics.

Let us start by looking at the following examples of context-dependent character identification: It obvious to a human reader that the symbol shown in Figure 2.a is "0", when it appears within an expression such as in Figure 2.c ($\sum_{i=\square}$). The context information assures us that this character cannot be a letter "o", "O", "omicron", the degree symbol $^\circ$ or the circle $\circ$. On the other hand, a recognition system that is not context-aware may equally well choose any of these candidates based on the best geometrical match. Similarly, a naïve automated recognizer might identify the character shown in Figure 2.b as "i". This would be perfectly acceptable when this symbol occurs in the expression $\sum_{\square=0}$, as shown in Figure 2.c. However, if the same character is encountered in the expression $\square = \frac{\partial z}{\partial t}$ (Figure 2.d), the recognizer should choose $\dot{z}$ as the best candidate.

In previous work [3, 9] we have presented a cross-application pen interface for recognizing handwritten mathematical expressions. Watt and Xie [5, 6] have studied methods to improve the performance of character recognition for large mathematical alphabets based on character *features*. In the present work, we explore techniques to further increase recognition accuracy by using the context information of mathematical expressions.



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

Figure 2. Ink samples: isolated and within context

(a) (b)

Figure 3. Spacial information could aid recognition

This paper is organized as follows: Section 2 describes our approach to data mining to obtain the mathematical analogue of a "vocabulary". Section 3 shows how the prediction information can be derived from the context of a mathematical expression. Section 4 reviews methods to combine prediction and recognition results and gives an overview on how prediction affects the accuracy of character recognition. Section 5 summarizes our conclusions and outlines some directions for future work in this area.

2 Building Mathematical Dictionaries

It is impossible to predict all the possible mathematical formulae that people may write. We can, however, determine a fixed set of the most common expressions currently used in practice by a population of research mathematicians. For this we need to find a reliable source of mathematical content that is neither writer- nor area-specific. Then we can use a collection of the most popular expressions as a "dictionary" for mathematics.

2.1 Data Mining

In earlier work Watt and So [4] have studied the most popular mathematical expressions used in practice. As part of this work, they have analyzed and categorized more than 20,000 mathematical documents from the mathematical arXiv service [1] for the period 2000-2004. We have used these results to create an initial dataset for building mathematical dictionaries.

2.1.1 Representation of mathematical content

We started by collecting all mathematical documents from the arXiv e-Print server that were classified under the AMS 2000 Mathematics Subject Classification and were accompanied by TEXsources. In total we were able to obtain more than 40GB of TEX sources.

Although TEX documents contain explicit encoding of mathematical notation, TEX markup typically does not represent the semantics, or even the grouping, of the formulae. We partially overcome this deficiency by translating mathematical content from the TEX format to Presentation MathML [2]. Using this standard we are able to represent the structure of formulæ as XML trees, which also implicitly encode semantics. As result, we obtained a 7GB collection of 250,000 MathML-encoded expressions.

2.1.2 Serializing formulæ

The theory of probabilistic prediction on strings is more straightforward than that for trees. Therefore, in order to compile a dictionary of mathematical expressions, we had to translate the two-dimensional structure of every formula in our collection to a linear sequence.

Obviously, the order in which two-dimensional formulæ can be traversed is not unique, and it is not clear which order is most used in handwritten expressions. Most typically, ambiguous situations arise with fractions and operators that take more than one script. For fractional expressions, we decided to adopt the order in which the numerator appears first, followed by the fraction bar and then the denominator. For the multi-script constructs we have established an order corresponding to the pattern ${}^{4}_{3}\square^{2}_{1}$. We are currently experimenting with a large test suite to determine whether these conventions will be suitable for most cases.

Our other concern was to preserve spatial relations between individual characters in the expression. For this purpose, we introduced four meta-tags: `<sub>`, `</sub>`, `<sup>`, `</sup>`, `<frac/>` and `<root/>`. By serializing expressions in this manner, dictionary methods are able to distinguish between the sequences encoding $y\,2$ and y^2. Assuming that the first expression is more popular than the second, a mathematical dictionary containing spatial information would be able to recommend the digit '2' as a candidate for the character circled in Figure 3.a, while suggesting the letter 'z' for the symbol circled in Figure 3.b

We used this method of expression serialization to convert all formulæ from our MathML collection to strings. Each string contained a combination of meta-tags and mathematical symbols in Unicode format. For example, the serialization process for the expression $\lim_{x\to 0}\frac{1}{x^2}$ gives the following sequence: `l, i, m, <sub>, x, →, 0, </sub>, 1, <frac/>, x, <sup>, 2 </sup>`.

2.2 Collecting Sequences

After the serialization stage, we obtained 250,000 sequences of mathematical literals and spatial tags. From these we have constructed all subsequences of length 3, 4 and 5. This allowed us to find common parts of formulæ that rarely appear on their own. For example, the expression ∂x^2 is almost never used as a stand-alone formula, but is often encountered as a denominator.

For each of the lengths 2–5 we obtained 403,000,000 subexpressions over an alphabet of 510 symbols (504 mathematical characters plus 6 spatial tags). Each of three collections was stored in a separate database, corresponding to the maximum entry length.

2.2.1 Sequence count

The collection of 403,000,000 subsequences contains many repeated entries. On average less then 2% of the subsequences do not have duplicates in the collection. To avoid storing redundant information, all repeated sequences are represented in the database as a single entry, accompanied by its *count* – the number of times the subexpression appears in the original collection.

2.2.2 Popular subexpressions

As expected, among subexpressions with high ranks there are commonly used formula fragments, such as $\sum_{i=1}$, $\sin\theta$, (x, y), $f(x)$, *etc*. As well as actual subexpressions, we also found popular *patterns* for formula structures, such as "$\square_{\square=1}^{n}$". This entry, for example, shows that if an operator such as $\sum$ or $\prod$ has the lower bound equal to 1, then the upper bound is likely to be n.

Some of the subsequences we found in the top 10% of database entries could have been foreseen. Others were less obvious. For example, the most popular subexpression of length 5 among the 38,000 sample mathematical documents is "^{- 1} (", which corresponds to the formula fragment "$\square^{-1}$(". This is why we believe it is important to obtain the entries for the mathematical dictionaries from empirical data, rather than from assertions based on intuition or "common knowledge".

2.3 Building Compact Dictionaries

Once we had populated the databases with subexpressions, we faced a new problem: Even without storing duplicates, the collections are too large. In the worst case, the size reached 660MB. This made the collections difficult to store and slow to access. Moreover, these databases were originally created to serve as mathematical dictionaries for assisting character recognizers. Given that recognizer systems are typically designed to work on portable devices such as the Tablet PCs and Pocket PCs, dictionaries of this size are not usable. We therefore had to find a criterion to reduce subexpression databases to a more reasonable size without sacrificing their useful content.

2.3.1 Expression distribution

Our first observation is that the popularity of an expression is represented by its count, *i.e.* expressions with low counts are not used frequently enough to warrant being included in a dictionary. The second observation is that as the count decreases, the number of seldom-used subsequences with that count increases, and there are few subexpressions that are very popular. The data we collected from the three databases showed similar dependency of number of different sequences of each counts on relative frequency. In each case the relation was monotonically decreasing, concave upward, similar to a negative exponential as can be seen in Figure 4.

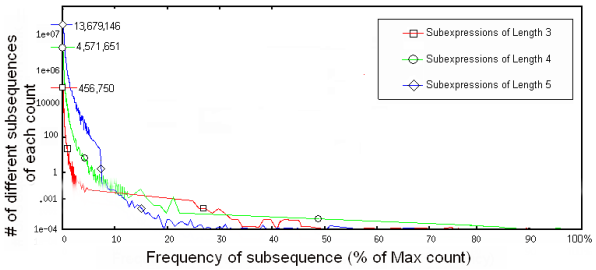


Figure 4. A diagram of distribution of mathematical subexpressions with respect to their frequency

Table 1. Three types of Compact Dictionaries for sequences of length 3-5

Type	Purpose	Min Expr. Count	Min. occurrence rate	# of different entries	Size
"Light"	Recognizers on PDAs	1000	0.00025%	30,000-40,000	0.3-0.5MB
"Practical"	Recognizers on TabletPC	400	0.0001%	80,000-200,000	1.0-3.4MB
"Detailed"	Experimental	100	0.000025%	220,000-400,000	10-22MB

2.3.2 Refining databases

The dependency in expression distribution suggests that by excluding the least popular expressions, we can significantly reduce the database size without reducing its utility. We therefore explored what would be a reasonable point at which to cut-off the database entries. For subexpressions of each length we created three databases, corresponding to different levels of completeness: "detailed", "medium" and "light". The qualitative characteristics for each type are summarized in Table 1.

2.4 Mathematical Context Databases

Dictionary-based methods for recognition of natural languages are based on word matching. This strategy will not work in mathematics: Due to the wide variety of mathematical expressions, dictionary support in pen mathematics should be based on matching arbitrary parts of the formulae, rather than whole expressions.

As described in Section 1, recognition of mathematical characters is often determined by the local context. We suggest the use of collections of subexpressions, as described in Section 2.2, for creating *mathematical context databases*. Thus, given a local spatial context[1] for a character in a handwritten formula, we will be able to determine whether a recognition candidate for this character would likely appear in an expression.

[1]Because our context databases are static we do not fully use information from the temporal context. Instead, the spatial context of each character is re-computed each time a new symbol is added to or deleted from the expression. This maintains consistency between the spatial and temporal contexts.

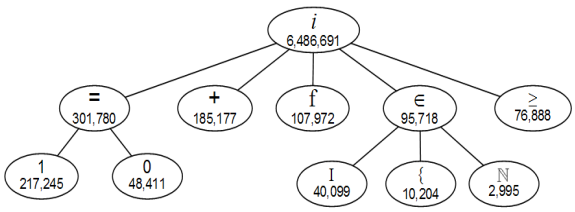


Figure 5. Part of a context database starting with i

Figure 6. Ambiguity resolvable by a context database

We shall call a length of the sequence encoding the preceding context as the *depth of the context*. As a part of mathematical expression, context is presented by two-dimensional structure; therefore, its encoding will contain meta-tags, that are also counted in total length of the encoding sequence. The *depth* of a context database is the maximum depth of its entries.

To date, we have built nine context databases, of various depths of context and levels of precision (see Section 2.3.2). Four of them are included in the mathematical recognition software Mathink, presented in [9].

2.4.1 Context and possible continuations

Each entry in a context database has two parts: *local context* and its *continuation*. Depending on the depth of the local context, the database can suggest a different number of continuations. For example, according to the fragment of a context database shown in Figure 5, a formula ending with "..i" may have five possible continuations, while an expression ending with "..$i \in$" may have only three: "I", "{" and "$\mathbb{N}$".

Symbols found among possible continuations of a particular context can advise recognition systems of the candidates that are more appropriate in a given context. Thus, for example, the context database would suggest that a hand-printed character on Figure 6 should be recognized as a capital letter I, and not as, for example, a number 1.

These ideas can be made more precise by considering the counts as defining a measure. Moreover, weights of words can be used to estimate a "likelihood" for the corresponding continuation. For example, according to the fragment in Figure 5, the letter "I" would be four times as likely to continue the sequence "$i \in$" as the open brace symbol, and thirteen times as likely as the symbol "$\mathbb{N}$." In the next sections we show in detail how to use this count information in cooperation with character recognition.

3 Character Prediction in Mathematical Expressions

As described in the previous section a context database suggests possible continuations for an input sequence. To do this we need to extract the local context from a partially-entered formula and match it against context database entries. In this section, we describe a technique to obtain the local spatial context from handwritten mathematical expressions. We also demonstrate by example how to use context information to calculate predictions for possible continuations.

3.1 Initial Settings

The context databases and the prediction methods are designed to be used for character prediction in mathematical handwriting recognition software. Usually, linear mathematical handwriting flows in one direction. The current version of our recognition system, Mathink, handles Western-style handwriting: from left to right. Furthermore, the design of our system assumes that every character is recognized immediately after it has been entered. This strategy ensures that by the time a new symbol is written, all previous characters have been recognized correctly, so we can use them as data in computing prefixes for further analysis.

These two observations imply that for our existing settings we can consider only context that *precedes* the recognized character. For this reason we have organized the context databases as prefix-defined, *i.e.* they provide easy access to *continuations* of a given sequence, but do not offer efficient methods to retrieve information about characters that may precede the context. In this paper we do not address the question of using the *surrounding context*. However, the approach of using the local surrounding context in global expression recognition appears to be promising.

3.2 Retrieving Context from Mathematical Expressions

We require a technique to identify the local spatial context of any character in a handwritten formula, regardless of whether this character has been recognized or just entered. For clarity, we demonstrate the method for context retrieval by example: Given the partially recognized formula of Figure 7.a, suppose the user has just added a new ink character below the horizontal bar. Now, *before* recognizing the new ink, the system has to identify in which spatial context this character appears. To answer this question, first, it must analyze and serialize the structure of the expression. From the resulting sequence of linearized expression structure it then needs to extract the part that corresponds to the neighborhood areas on the left, above or sometimes below the handwritten character.

(a) (b)

Figure 7. Partially recognized expression $\frac{\sqrt{3}}{\cdot}$ (a) and a corresponding layout tree (b)

To determine the structure of the expression, including the newly entered ink, we cannot use a regular structure analyzer, since it requires the character to be recognized first. On the other hand, we cannot leave the new entry out while computing the context. For instance, in our example the final result depends on the fact that a new character is written *under* the rest of the expression. To resolve this situation, we use a preliminary structure recognizer: a "layout analyzer" that estimates the role of a new entry in the formula structure, by using only the size and the position of its bounding box. Thus, the expression from Figure 7.a will be translated into a layout tree, presented on Figure 7.b.

The layout tree can be easily converted into a linear sequence, using the serializing methods presented in Section 2.1.2. For example, the tree in Figure 7.b would be linearized as <root/> 3 <frac/>. Then, the context is defined by the last $n-1$ literals of this sequence, where n is the depth of the context database. So, if we use a context database of depth more than 4, we can send the whole sequence to the next stage for computing potential continuations.

3.3 Calculating Character Prediction

To compute possible continuations of a given context, we send a query to the context database. If this succeeds, we get the result as a weighted vector of alternatives, which we can normalize and export as a set of ranked candidates for character predictions. If the current context of length l is not found in the context database, we try again with a subsequence of the last $l-1$ literals and repeat the procedure.

From the perspective of expression analysis, possible continuations of a given context can be considered as *prediction candidates* for a newly entered character. Then the normalized rank of each candidate can be used as a *prediction confidence* in further analysis. In the next section we show how prediction information may affect the results of isolated character recognition.

4 Combining Prediction and Recognition

We are interested in whether prediction results can improve recognition. There are many ways one may use a ranked list of predicted next character possibilities. At a minimum, the list of likely continuations could be used as a filter to prune the search space for a recognizer. Knowing likely next characters could also be used to select the most strongly separating features for those characters and advise the recognizer to budget more time in analyzing those. The ranking of the most likely next characters could also be used to re-order closely ranked recognition results. It is this last possibility we explore here.

Table 2. Ranking of recognition results for the handprinted character in Figure 7.a

Char	Recognition Confidence	Original Rank	Prediction Confidence	Combined Confidence	New Rank
Z	0.912	1	0	0.912	2 (↓)
z	0.841	2	0	0.841	3 (↓)
2	0.61	3	1	1.61	1 (↑)
≀	0.102	4	0	0.102	4
⊇	0.034	5	0	0.034	5

Table 2 presents an example of combining confidence scores, where the correct candidate for the denominator in Figure 7.a is "promoted" owing to its prediction confidence. In practice we commonly face situations where prediction and recognition results contradict each other. In these cases we have to decide how to combine the confidence values of character prediction with those of recognition.

There have been many suggested methods to combine methods that give different results. The work of Kuncheva[7] or Kittler [8], for example, serve as a useful starting point for those unfamiliar with this literature. For simplicity, in this preliminary study, we explore combining scores using simple arithmetic models. When these produce significant results, more sophisticated methods can be used.

4.1 Combination Functions

If we define the final rank of a candidate as a function of two arguments: *recognition certainty* $\mathcal{R}$ and *prediction confidence* $\mathcal{P}$, then we need to find a reliable method to combine these values. Regardless of the type, the combination function $\mathfrak{C}$ shall have two parameters. These are weight coefficients for recognition and prediction confidences. For concreteness, in this work we have considered three types of combination functions $\mathfrak{C}(\alpha, \beta)$:

$$\mathfrak{C}_1(\alpha,\beta) = \alpha \cdot \mathcal{R} + \beta \cdot \mathcal{P} \quad (1)$$

$$\mathfrak{C}_2(\alpha,\beta) = \mathcal{R}^{\alpha} \cdot \mathcal{P}^{\beta} \quad (2)$$

$$\mathfrak{C}_3(\alpha,\beta) = \alpha \cdot \exp(\mathcal{R}) + \beta \cdot \exp(\mathcal{P}) \quad (3)$$

Within this setting our goal is to estimate the best combination of α and β by maximizing $\mathfrak{C}_i$ over a set of handwriting samples. We must avoid choosing the coefficient based on the results of too few experiments with single users' handwriting. Unfortunately that approach to handwriting analysis is too often practiced by system developers. The approach we took is instead based on the use of

empirical data obtained from larger test suites. Although in our experiments we used prototype test sets, the approach we have developed can be extended to larger collections to obtain more general results.

4.2 A Testbed

We chose to search for a combination of the coefficients α and β that optimized the recognition rate for a fixed set of handwritten expressions. To do this, we have built a test suite that had used seventeen representative formulæ from different areas of mathematics. We then collected handwritten samples of these formulæ from 10 different writers.

To analyze the experimental data, we have developed a software environment for testing based on the Mathink recognition system [9]. Once the handwritten samples were collected, we ran the Mathink recognizer on a set of 170 expressions, using the grid $[1..20] \times [0..10]$ for α and β, repeating calculations for each of the three combination formulae.

4.3 Experimental Results

The experiments have shown that there is always a combination $\langle \alpha, \beta \rangle$ that increases recognition accuracy for each writer individually, for every formula and, even better, for an average of all individual writers and formulas. The experiments have also confirmed that heavily relying on prediction, when recognition confidence is low will significantly decrease the overall recognition rate, in the worst case by up to 10%.

The diagrams in Figures 8, 10, 12 represent typical results of the test cases for $\mathfrak{C}_1$, $\mathfrak{C}_2$ and $\mathfrak{C}_3$. They show the increase in accuracy compared to the results of character recognition, obtained without using prediction information. The overall distribution of the coefficients α and β yields the patterns of Figures 9, 11, 13.

Our search for good values for α and β has been based on a naïve evaluation of points on a grid. More sophisticated optimization strategies should prove both more efficient and yield better results.

5 Conclusion

We have explored methods to use the context to improve on-line mathematical handwriting analysis. While dictionary-based methods are widely practiced in handwriting recognition for natural languages, there is no fixed vocabulary to support them for mathematical content. Nevertheless, in practice some mathematical expressions are used much more often than others. This observation allows us to define a measure on the set of expressions and use this in place of a dictionary.

In this work we have discussed how context databases can be built from sets of mathematical expressions used in practice. We have also shown how we derive the local context from the partial structure of mathematical formulæ and have demonstrated how this context information can be matched against context databases.

Furthermore, we have investigated methods of combining prediction information with recognition results. We have built a framework based on the Mathink system to validate our approach. We used this testing environment to drive an experimental study designated to determine the best combination of context prediction and character recognition. The experimental results confirmed the existence of a combination method, increasing the accuracy of isolated character recognizers.

Our on-going research examines a variety of strategies for such combinations, on verifying these results against larger writing samples, and on using context databases specific to particular areas of mathematics.

References

[1] ArXiv e-Print Archive, `http://arxiv.org`, (c) 2000-2004.

[2] R. Ausbrooks, S. Buswell, D. Carlisle, et al., *Mathematical Markup Language (MathML)* version 2.0 (second edition), World Wide Web Consortium, 2003.

[3] Elena Smirnova and Stephen M. Watt, "A context for pen-based computing", *Proceedings of the Maple Conference 2005*, Maplesoft, 2005, pp. 409–422.

[4] Clare M. So and Stephen M. Watt, "Determining empirical properties of mathematical expression use", *Proceedings of Fourth International Conference on Mathematical Knowledge Management, (MKM 2005)*, Springer Verlag, 2006, pp. 361–375.

[5] Stephen M. Watt and Xiaofang Xie, "Prototype pruning by feature extraction for handwritten mathematical symbol recognition", *Proceedings of Maple Conference 2005*, Maplesoft, 2005, pp. 423–437.

[6] Stephen M. Watt and Xiaofang Xie, "Recognition for large sets of handwritten mathematical symbols", *Proceedings of IEEE International Conference on Document Analysis and Recognition (ICDAR 2005)*, vol. 2, 2005, pp. 740–744.

[7] Ludmila I. Kuncheva, "Combining Pattern Classifiers: Methods and Algorithms", ISBN: 978-0-471-21078-8 Hardcover, 376 pages, July 2004

[8] Josef Kittler, Mohamad Hatef, Robert P.W. Duin, Jiri Matas, "On Combining Classifiers", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, Issue 3 (March 1998) ISSN:0162-8828, pp. 226-239

[9] Elena Smirnova and Stephen M. Watt, "A cross-application architecture for pen-based mathematical interfaces", *Proceedings of Mathematical User Interfaces (MathUI 2007)*, RISC, Austria, 2007 `http://www.activemath.org/workshops/MathUI/07/proceedings/Smirnova-Watt-MathInk-MathUI07.pdf`.

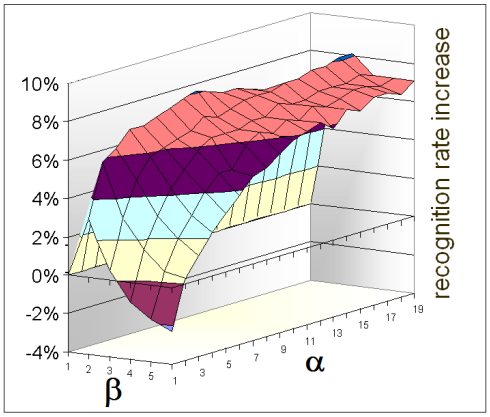


Figure 8. Results for user 1. Best improvement for $\mathfrak{C}_1$ is 8.1% at (9,1).

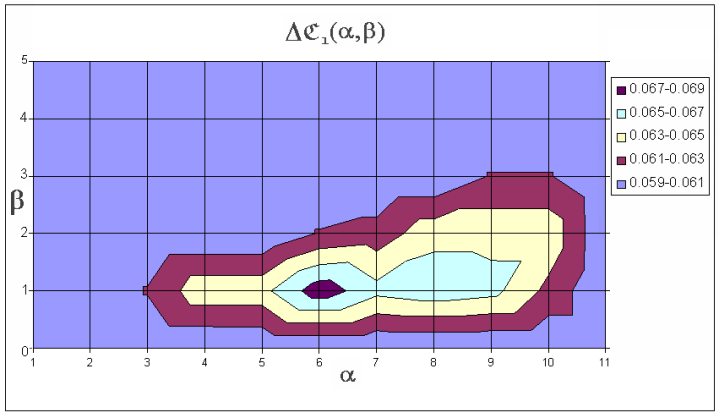


Figure 9. Improvement in $\mathfrak{C}_1$ as a function of α and β

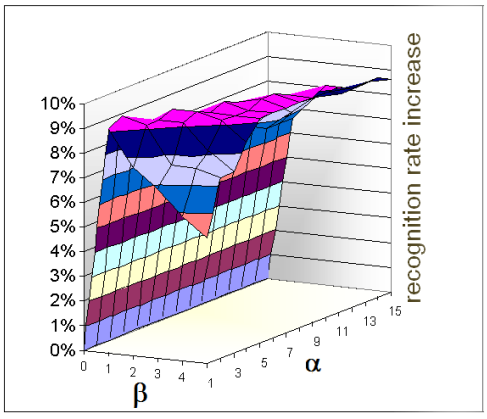


Figure 10. Results for user 2. Best improvement for $\mathfrak{C}_2$is 9.5% at (2,1), (4,3) and (7,4).

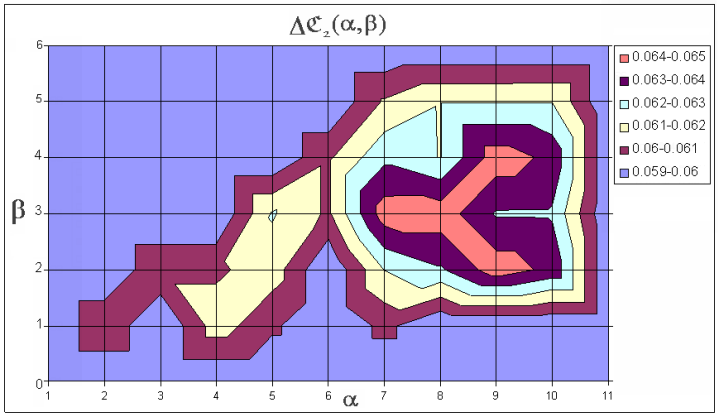


Figure 11. Improvement in $\mathfrak{C}_2$ as a function of α and β

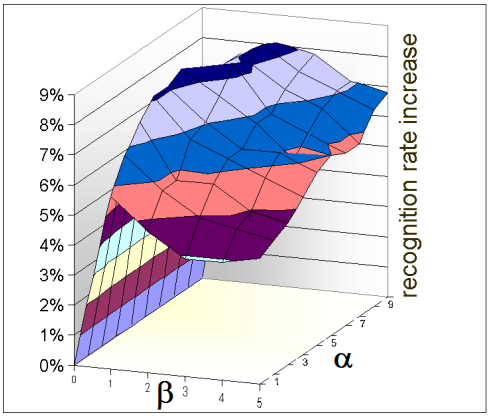


Figure 12. Results for user 3. Best improvement for $\mathfrak{C}_3$ is 7.5% at (6,1), (7,2) and (9,2).

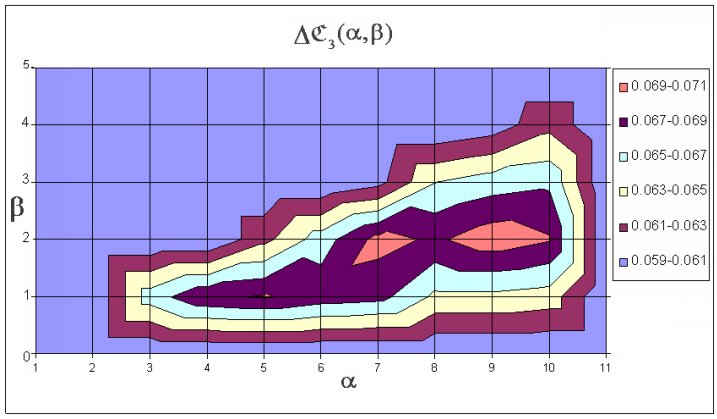


Figure 13. Improvement in $\mathfrak{C}_3$ as a function of α and β