RECOGNITION OF ON-LINE HANDWRITTEN FORMULAS

A. KOSMALA, G. RIGOLL
Faculty of Electrical Engineering - Computer Science
Gerhard-Mercator-University Duisburg
47057 Duisburg, Germany
E-mail: {*kosmala, rigoll*}*@fb9-ti.uni-duisburg.de*

This paper describes a system for the recognition of online handwritten mathematical expressions. The system presented here is based on Hidden Markov Models (HMMs) and offers thus the advantage of a simultaneous segmentation and recognition, avoiding the complex and crucial handwriting segmentation during pre-processing. The segmentation and recognition result is used in a further step for the interpretation of the symbols and their spatial relationships. Under consideration of some constraints to the handwriting production process, ambiguities can be resolved and the results are transformed into TEX-syntax for visualization purposes.

1 Introduction

Assuming a robust automatic recognition of mathematical expressions, handwriting could be a superior man machine interface for the input of formulas and scientific documents. Most of the word processors as well as mathematical programs offer only a poor input interface for mathematical expressions. The automatic processing and recognition of handwritten mathematical notations is a challenging task, requiring solutions to two complex problems[1]:

1. The segmentation and recognition problem of the distinct symbols
2. Analysis of the two-dimensional symbol arrangement, with an interpretation of the relative symbol positions and relative symbol sizes.

While the human recognition of mathematical expressions is supposed to be a more embedded process of recognition and understanding, it is convenient for the development of an artificial recognition system to divide the entire process into the above mentioned two problems. As documented in the overview of Blostein[1], all mathematics-recognition systems are based on this two-step architecture.
The first main processing step can be subdivided for most of the discussed implementations into three further steps, like noise reduction, segmentation and symbol recognition. Instead of an explicit segmentation of the single symbols, the system presented here makes use of the simultaneous segmentation and recognition capabilities of HMMs. Furthermore, HMMs offer superior learning capabilities and allows thus a fast adaptation to new writers. In contrast to the HMM-approach presented in [2], where each distinct feature stream is classified separately, the approach presented in this paper applies the robust multi-stream recognition with the methods presented

in [3,4,5]. This approach allows a simple extraction of geometrical features from the segmentation / recognition result for the subsequent structural analysis.
To avoid the introduction of a two dimensional grammar, some constraints are presented, which allow a sequential analysis of the transcription, considering a natural style of handwriting.
The following section gives an overview of the entire system with a description of the training- and test-set, and the constraints to the handwriting production process. The third section describes each module of the first main step more detailed, starting with the pre-processing, feature extraction and the Hidden Markov modeling. The fourth section describes the structural analysis of the recognition result, before the results and conclusion will be presented in the fifth and sixth section.

2 System Outline

The system proposed here is capable of recognizing mathematical expressions with a set of 100 different characters or symbols in a writer dependent mode. Beside small and capital letters, and digits, the system contains several mathematical symbols ($+ - \cdot : / \overline{} \hat{} \sqrt{\ } \sum \prod \int ,' = < > \leq \geq \rightarrow \leftrightarrow \approx ! \infty$), as well as some of the most frequently used Greek letters ($\alpha, \beta, \gamma, \lambda, \mu, \Delta, \pi, \omega, \epsilon, \tau, \phi$) and some parentheses ((,), [,], {, }). To model the spaces between the symbols, an additional 'space' model is introduced. For initialization, each of these symbols is written several times well separated on a sheet. For an embedded training, 100 common mathematical or physical formulas are collected as training set and an additional set of 30 formulas is used as test set, represented by 150000 vectors and 45000 vectors, respectively.
With respect to a natural writing style, some constraints are introduced to simplify the subsequent structural analysis:

1. In general it is required, that the expressions have to be written from left to right and from top to bottom. In case of a fraction this means, that first the numerator has to be finished, than the fraction line has to be drawn, and finally the denominator has to be written, which is anyway most frequently the case. This means further, that expressions with any kind of parentheses have to be written in the temporal order '*opening parenthesis, expression, closing parenthesis*' and not as it is sometimes desired in the sequence '*expression, opening parenthesis, closing parenthesis*' resulting in the same spatial order.

2. It can be observed, that in case of writing sums, integrals and products the writer usually writes first the lower limit and after this the upper limit. A similar situation can be found when subscripts occur in combination with superscripts, quotes or vector- and matrix-arrows, respectively. Here also the writer usually

writes, subsequent to the in-line symbol, first the subscript and then the upper section symbols, like superscripts etc. Thus, this constraint is an exception of the first constraint, because in these cases the top to bottom constraint from (1) is resolved.

3. Regarding operators like fractions, or roots, it might be useful in some cases to expand the fraction line or the bar of the root symbol if the denominator or radix gets longer than expected before. In this case it is due to the use of dynamic information not possible to expand the line afterwards.

4. Currently, no higher order root operations, no operators as subscript and no higher level superscripts are allowed like x^{y^z}, for example.

Fig. 1 gives an overview of the complete system. The processing levels as they are shown, will be described in the following sections.

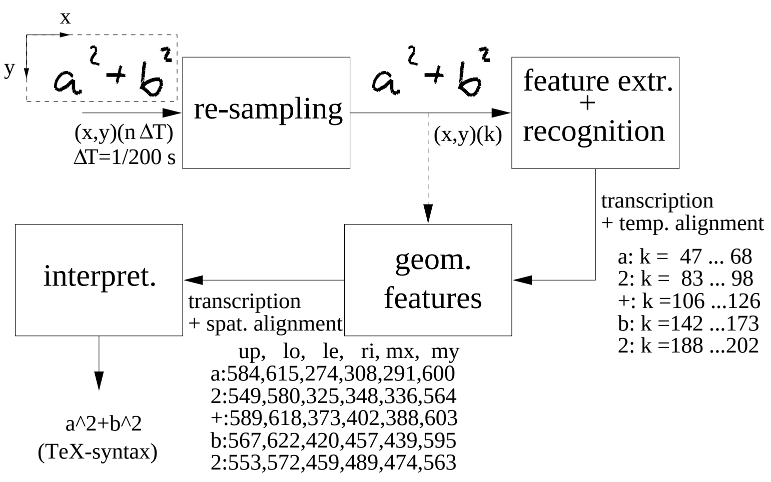


Figure 1: System overview

3 Formula Recognition

Considering the constraints in Sect. 2, it is not necessary to enter each single symbol well separated, neither spatially nor temporally. This continuous formula recognition can be considered as a kind of sentence recognition without previous word segmentation. The segmentation of the distinct words is realized within the HMM framework, applying a special 'space' HMM for the detection of word boundaries. Instead of attempting to identify each symbol separately, the efficient HMM-based decoding techniques allow a complete identification of the entire formula string. Another advantage of the HMM approach for this application is the possibility to incorporate high level syntactic constraints (e.g. grammars) directly into the low level recognition process.

3.1 Pre-Processing and Low-Level Feature Extraction

The raw data is captured with a constant sample rate of 200 Hz, resulting in a temporal vector sequence of the Cartesian coordinates of the pen position. The pre-processing step is a re-sampling of the captured pen trajectory with vectors of constant length [3]. Beside the data reduction effects of 1:2 up to 1:3, a further advantage of the re-sampling is, that the implicitly given writing speed, resulting from different distances between two samples and the constant sample rate, is eliminated. Writing speed is especially in the context of formula recognition supposed as a highly inconsistent feature, because obviously identical handwriting images can be produced with completely different writing speeds. For instance, this could happen if the writer holds on to think about his currently written formula. It should be stressed, that the re-sampling preserves the spatial information, i.e. the Cartesian pen coordinates, as shown in Fig. 1.
On the recognition level, several features are extracted from the re-sampled vector sequence. The first type of features are the online features [3], which is the orientation α of the re-sampling vectors coded in sine and cosine as well as the sine and cosine of the differential angle $\Delta\alpha$ of two successive re-sampling vectors. The third online feature is the pen pressure, which is extracted as a binary feature and indicates if the pen is set down or lifted. The pen pressure is helpful to model word or symbol boundaries within an expression. The second type of feature is an off-line feature [5,6], which is a sub-sampled bitmap, slided along the pen-trajectory after re-sampling. In a subsequent step, each of these feature streams, except the binary pressure, is quantized with a k-means vector quantizer with different codebook sizes for each feature. This resulting discrete multi-stream is presented to the HMMs.

3.2 HMM Training and Recognition

For the modeling of the symbols, discrete left to right HMMs without skips and with different numbers of states are used. With the features described above, discrete HMMs have shown to be superior compared to continuous HMMs [3]. One reason for this is the discrete nature of the pen-pressure (pen up or pen down). Furthermore, the α and $\Delta\alpha$-features are due to the constant re-sampling vector length distributed on the standard circle, which enables an efficient vector quantization, while Gaussian pdfs as they are commonly used in continuous HMM systems can not be sufficiently

mapped to this kind of distribution. HMMs with 12 states are used for capital letters and larger mathematical operators, like sum or product. Small letters and smaller operators are modeled with HMMs with 8 states, while very short symbols (.,) are modeled with 3 state HMMs.
For recognition purposes, a synchronous Viterbi decoding is used [7]. By means of the Viterbi algorithm, it is possible to determine the most likely state sequence $\mathbf{q}^\star$ of a set of HMMs λ for a given sequence of frames $\mathbf{O}$.

$$P(\mathbf{O}, \mathbf{q}^\star|\lambda) = \max_{\mathbf{q}} P(\mathbf{O}, \mathbf{q}|\lambda) \tag{1}$$

This can be very effectively exploited for formula recognition by analyzing the resulting alignment of each feature frame to its best matching HMM state, with the result, that the indexes of start- and end-frames of the recognized symbols within an expression can be taken directly from the decoding or recognition process. The achieved recognition with the simultaneous segmentation of the symbols is an important information for further extraction of geometrical features for the structure analysis.
The initialization of the HMMs is realized with the Viterbi training using the separate collected, isolated samples of the symbols. HMM parameter optimization is carried out by an iterative application of the Viterbi algorithm in order to find an optimal state sequence, and a re-estimation path of the pdfs.
For the embedded training of the complete formulas, the Forward Backward algorithm is used. Again, in an iterative way, optimized HMM parameters $\hat{\lambda}$ can be found by a maximization of the *Kullback-Leibler distance* $Q(\lambda, \hat{\lambda})$:

$$\max_{\hat{\lambda}} Q(\lambda, \hat{\lambda}) = \max_{\hat{\lambda}} \sum_{\mathbf{q}} P(\mathbf{q}|\mathbf{O}, \lambda) \log P(\mathbf{O}, \mathbf{q}|\hat{\lambda}) \tag{2}$$

The re-estimation formulas for the HMM parameter set can be derived directly from $Q(\lambda, \hat{\lambda})$ [7].

4 Structure Analysis and Synthesis

As described before, the result from the Viterbi decoder is not only the transcription, i.e. the sequence of recognized words and symbols, but also the start- and end-frames of each symbol. The following sub-sections describe, how this information can be used for a further processing.

4.1 Extracting Geometrical Features

The temporal alignment of the frames together with the re-sampled vector-data allows the extraction of geometrical features, i.e. a spatial alignment of the symbols. These geometrical features are the center of a recognized symbol and the size and position of its bounding box. As shown in Fig. 2, the bounding box of a word is extracted by stepping through the re-sampled vector sequence starting at the start-frame number (k=35) of the current word until the stop-frame number (k=59) is reached, and searching for minima and maxima in x- and y-direction in the determined part of the sequence of Cartesian vectors. The center of the word or symbol (mx, my) is approximated with the geometrical center of its bounding box. The spatial alignment for each symbol is characterized by the parameters of the bounding box up, lo, le, ri and denotes the upper, lower, left and right boundary of the symbol. The centers of the bounding boxes mx and my of each symbol are shown in the last two columns of the spatial alignment in Fig. 1 and Fig. 2.

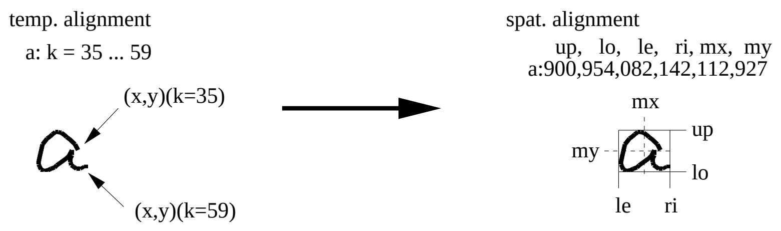
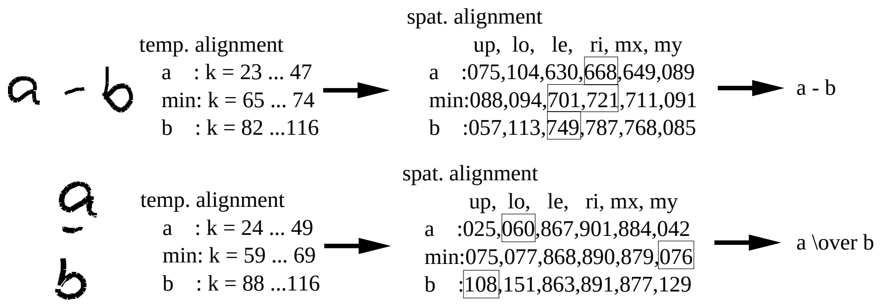


Figure 2: Spatial- from temporal alignment

4.2 Interpretation of the Geometrical Structure

With the spatial alignment the sizes of the words and their relative positions are known. Furthermore, with the constraints as mentioned in Sect. 2 it is also known, in which temporal direction expressions with special meanings, like the limits of an integral have to be searched or how to resolve ambiguous recognition results. This is only possible with the generated additional size- and position-information.

Recognition Error Correction The first interpretation step of the spatial alignment is the correction of obvious recognition errors. This can be a possible confusion between minus and short fraction lines (Fig. 3) or quotes and commas or an inserted '$\cdot$' after the words sin or lim caused by a possible delayed stroke of the subsequent set i-dot. While the inserted '$\cdot$' after the words sin and lim is deleted in general, because it does not form a valid expression, the confusion between minus and short fraction is resolved by comparing the relative position of the dash to the temporal predecessor symbol (a) and to the successor symbol (b). If the center of the dash $(my(min))$ is below the lower bound of the predecessor symbol $(lo(a))$ and above the upper bound of the successor ($up(b)$), then the dash line must be a fraction, otherwise it must be a minus. A similar procedure is started with the comparison of the relative positions of predecessor and successor, if either a quote or a comma is found in the transcription.



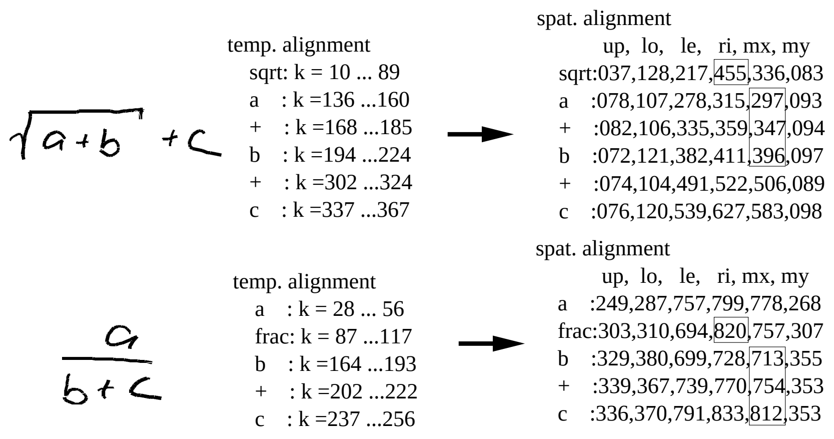
```
                                   spat. alignment
          temp. alignment               up,  lo,   le,   ri, mx, my
a - b       a   : k = 23 ... 47    a   :075,104,630,[668],649,089
            min: k = 65 ... 74  →  min:088,094,[701,721],711,091   →  a - b
            b   : k = 82 ...116    b   :057,113,[749],787,768,085

                                   spat. alignment
 a        temp. alignment               up,  lo,   le,   ri, mx, my
 -          a   : k = 24 ... 49    a   :025,[060],867,901,884,042
 b          min: k = 59 ... 69  →  min:075,077,868,890,879,[076]   →  a \over b
            b   : k = 88 ...116    b   :[108],151,863,891,877,129
```

Figure 3: Sample for fraction/minus-confusion

Special mathematical operators This second step searches the transcription for special symbols like $\int$, $\sum$, $\prod$ and lim, with context words of special meaning. In case of $\int$, $\sum$, or $\prod$ an upper and lower limit has to be found. It is known from constraint (2), that the successor(s) of the word $\int$ for example has to be the lower limit, while the second next successor has to be the upper limit of the integral, as long as these successors fulfill the geometrical conditions, namely that they are below and above the integral, respectively. If no successors are found, that fulfill these conditions, the integral is an undetermined integral. A similar procedure is started if the word lim is found in the transcription. Then a search is started for the temporal successive argument of the word lim, which has to be located below the lim. If the word $\sqrt{}$ is found, the successors are grouped as the radix, as long as their centers are within the bounding box of the word $\sqrt{}$ (Fig. 4). The last type of this special mathematical operators are the fractions. Again constraint (1) defines the temporal order of numerator, fraction line and denominator, which means, that only previous words in the transcription can belong to the numerator and that only following words may belong to the denominator. Now only a decision about the beginning of the numerator and the end of the denominator is required. A preceding (successing) word is grouped to the numerator (denominator), if the center of this word is located on the right (left) of the left (right) edge of the bounding box of the fraction line.

Subscript and superscript Determining if a word is the base for sub- or superscript is the most complicated step in the structure analysis, because there is no hint avail-



```
                                        spat. alignment
                temp. alignment              up,  lo,   le,   ri, mx, my
                  sqrt: k = 10 ... 89      sqrt:037,128,217,455,336,083
                  a   : k =136 ...160      a   :078,107,278,315,297,093
                  +   : k =168 ...185      +   :082,106,335,359,347,094
                  b   : k =194 ...224      b   :072,121,382,411,396,097
                  +   : k =302 ...324      +   :074,104,491,522,506,089
                  c   : k =337 ...367      c   :076,120,539,627,583,098

                                        spat. alignment
                temp. alignment              up,  lo,   le,   ri, mx, my
                  a   : k = 28 ... 56      a   :249,287,757,799,778,268
                  frac: k = 87 ...117      frac:303,310,694,820,757,307
                  b   : k =164 ...193      b   :329,380,699,728,713,355
                  +   : k =202 ...222      +   :339,367,739,770,754,353
                  c   : k =237 ...256      c   :336,370,791,833,812,353
```

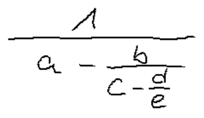
Figure 4: Grouping of sub-expressions

able, which enables a search for special key-words as it was the case in the previous steps. In fact nearly every character could be the base for sub- or superscript. Except some mathematical operators like fraction line or '$+$' for instance, which may not belong to a subscript. Thus the search for sub- or superscripts is just based on the relative positions of the words. Distinguishing between sub-, superscript and in-line is realized with the comparison of the word centers. An occurring subscript should be found successive to the in-line word located on the lower right, while an occurring superscript should appear after the subscript (if present) and should be located on the upper right. Some confusions can be observed, if descenders appear like g, j, p etc. Then it might happen, that a successive in-line word is classified as a superscript. To avoid this, a bias is added to the centers of such descenders.
In a similar way, the meaning of the symbols '$\rightarrow$' and '$\leftrightarrow$' is determined, if they represent a mathematical operation or a vector- or matrix-arrow.

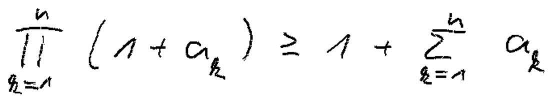
5 Results

On the writer dependent task as described in Sect. 2, we obtained a symbol recognition rate of 96.3% after the Viterbi recognition without further error correction. After the error correction, based on geometrical features, the recognition rate increases up to 97.7%. This shows, that the modeling works robust and that a confusion between short fractions and minus or between quotes and commas is not very frequent. A further source of errors is the similarity between o, O and 0 or between z and Z for instance. The differences between those symbols are even harder to detect, if they appear as sub- or superscript, because of the limited size variability. Recent exper-

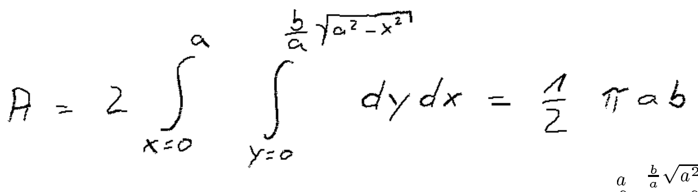
iments show, that these confusions could possibly be reduced with an increased accurate modeling, as it could be realized with the introduction of context dependent models[4] or with an additional MMI training of the vector quantizer codebooks[8,9].
The samples in Fig. 5 should give an idea of the type of formulas, which are used for the test. For the syntactic representation of the recognized formulas we chose the TEX-syntax, with the advantage, that the recognition results can be piped immediately to the TEX-compiler and can thus be visualized.



$$\Rightarrow \frac{1}{a-\frac{b}{c-\frac{d}{e}}}$$



$$\Rightarrow \prod_{k=1}^{n} (1+a_k) \geq 1 + \sum_{k=1}^{n} a_k$$



$$\Rightarrow A = 2 \int\limits_{x=0}^{a} \int\limits_{y=0}^{\frac{b}{a}\sqrt{a^2-x^2}} dydx = \frac{1}{2}\pi ab$$

Figure 5: Samples for correct recognition results from the test-set

6 Conclusion

The results presented in this paper show, that with the introduction of appropriate constraints a robust online handwritten formula recognition is feasible, even without requiring an artificial style of handwriting. Furthermore it is very helpful to perform segmentation and recognition in a single path in order to achieve a robust recognition, instead of applying segmentation algorithms in a pre-processing step.
Beside the investigation of the system for further writers and an enlarged set of symbols, future work will be focused on an integration of recognition and higher level analysis processes. Furthermore, some detail problems have to be solved, like the processing of higher order superscripts, for instance.

References

1. Dorothea Blostein and Ann Grbavec. Recognition of Mathematical Notation. In H. Bunke and P. S. P. Wang, editors, *Handbook of Character Recognition and Document Image Analysis*, chapter 21, pages 557–582. World Scientific Publishing, 1997.
2. Hans-Jürgen Winkler. HMM-Based Handwritten Symbol Recognition Using On-Line and Off-Line Features. In *Proc. IEEE Int. Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 6, pages 3438–3441, Atlanta, 1996.
3. G. Rigoll, A. Kosmala, J. Rottland, and Ch. Neukirchen. A Comparison between Continuous and Discrete Density Hidden Markov Models for Cursive Handwriting Recognition. In *Proc. Int. Conference on Pattern Recognition (ICPR)*, volume 2, pages 205–209, Vienna, 1996.
4. A. Kosmala, J. Rottland, and G. Rigoll. Improved On-Line Handwriting Recognition Using Context Dependent Hidden Markov Models. In *Proc. Int. Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 641–644, Ulm, 1997.
5. A. Kosmala, J. Rottland, and G. Rigoll. An Investigation of the Use of Trigraphs for Large Vocabulary Cursive Handwriting Recognition. In *Proc. IEEE Int. Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3373–3376, Munich, 1997.
6. S. Manke, M. Finke, and A. Waibel. Combining Bitmaps with Dynamic Writing Information for On-Line Handwriting Recognition. In *Proc. Int. Conference on Pattern Recognition (ICPR)*, pages 596–598, Jerusalem, 1994.
7. Lawrence R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. of the IEEE*, 77(2):257–285, 1989.
8. G. Rigoll. Maximum Mutual Information Neural Networks for Hybrid Connectionist-HMM Speech Recognition Systems. *IEEE Transactions on Speech and Audio Processing, Special Issue on Neural Networks for Speech*, 2(1):175–184, January 1994.
9. Gerhard Rigoll and Andreas Kosmala. An Investigation of Context-Dependent and Hybrid Modeling Techniques for Very Large Vocabulary On-Line Cursive Handwriting Recognition. In *6th International Workshop on Frontiers in Handwriting Recognition (IWFHR)*, Taejon, Korea, 1998.