AN INTUITIVE PEN-GESTURAL INTERFACE FOR SYNTACTIC-SEMANTIC ANNOTATION OF NON-CURSIVE HANDWRITTEN INPUT

Jörg Hunsinger

Institute for Human-Machine Communication
Technical University of Munich, D-80290 Munich, Germany
hunsinger@ei.tum.de

ABSTRACT

Any system designed for handwriting or drawing recognition, especially as far as probabilistic decoding components are involved, must be extensively trained by means of manually acquired, segmented, and syntactic-semantically annotated reference material. In order to standardize and facilitate these error-prone operations, we developed a structured graphical interface which utilizes simple pen gesturing as a highly efficient technique for graphical input manipulation. In particular, we use this tool to acquire and annotate training corpora of handwritten mathematical formulas. By applying corresponding feature extraction and iterative training mechanisms, the resulting set of probabilistic parameters is then fed into our single-stage top-down semantic decoder for formula understanding. Nevertheless, the interface is suitable for any other non-cursive handwriting or drawing application. On the signal near levels, the derived feature vectors may be processed by any handwritten symbol recognition stage.

1. INTRODUCTION

Our interface was especially designed for a pen operated LCD digitizing tablet providing instantaneous visual feedback. For enhanced interactivity we set a high value on on-line visualization features as well as back and forth switching facilities between the different involved subtasks. All the necessary worksteps require merely simple pen gesturing and/or pen selection events. The software is platform independent due to the use of the Tcl/Tk scripting language [1]. In the following section we give a detailed description of the relevant interface functionality.

2. METHODOLOGY

The main menu serves to switch freely between four basic workspaces:

2.1 Acquisition Workspace

Any new samples may be written down with arbitrary positioning on a scrollable area. Pen movements are sampled at 60 Hz so that sequences of handwriting strokes (i.e. segments between a pen-down and a subsequent pen-up event) successively appear on screen. All the collected data are stored online in an internal binary format. To keep the temporal order intact, which is implicitly evaluated in our decoding approach, only strokewise backward deletions, i.e. repeated undo operations, are allowed. Fig. 1 shows a typical handwriting sample being entered in this workspace.

2.2 Segmentation Workspace

After one or more handwriting or drawing samples have been collected, their piecewise segmentation to syntactic units (i.e. handwritten symbols or drawing elements, respectively) takes place in a separate workspace. The purpose of this stage is to group strokes belonging together by performing erase- or encircle-type pen gestures. Special color coding methods and - optionally - surrounding rectangles around every segmented stroke group (or single stroke, respectively) indicate segmentation consistency at a glance. During pen gesture formation, all the captured strokes are promptly highlighted to ease gesture execution and to visualize segmentation success incrementally. Symbol fragmentation is possible via the definition of two or more symbolic components assigned to a single semantic unit (cf. Sec. 2.4). In the mathematics domain, such a strategy makes sense for supporting e.g. a radical sign drawn in two parts, interrupted by a radicand expression. The segmentation scenario is illustrated in Fig. 2.

From: HCI 2001, Lawrence Erlbaum Ass., NJ

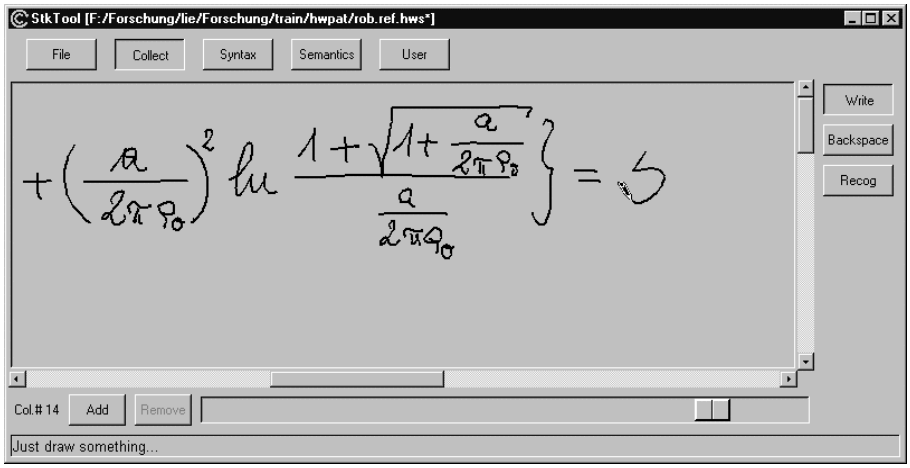

Figure 1: Acquisition Workspace. The lower horizontal scrollbar serves to browse the set of already acquired handwriting samples ("collections"). Via the Backspace button the user may delete one or more handwriting strokes in backward order.

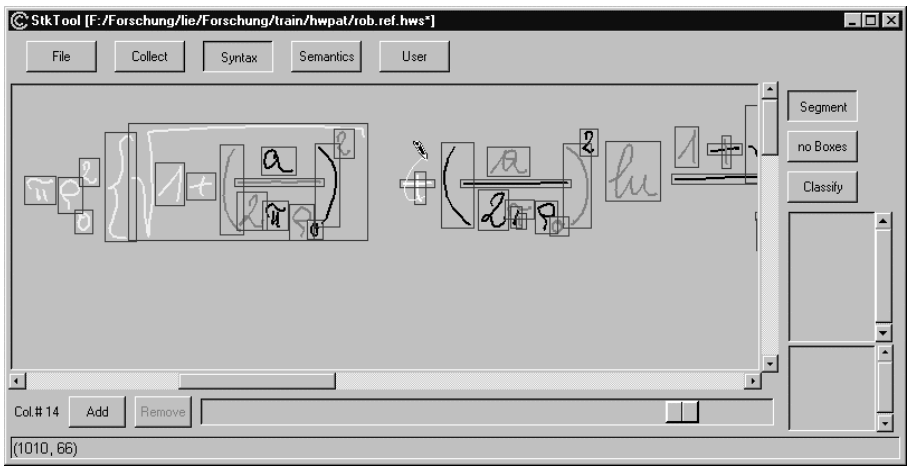

Figure 2: Segmentation Workspace. Initially, all the acquired handwriting strokes appear in alternating color according to writing order (here: black and dark grey in turn). During pen gesture formation, the captured strokes take on a different color (here: white), and will show a single bounding box around them afterwards. The pen gesture trajectory is displayed as a thin curve (here: white) during execution. Alternate coloring is also maintained between the final segmented stroke groups analogically. Stroke groups that were already syntactically and/or semantically annotated (cf. below) are indicated by another uniform color (here: light grey).

2.3 Syntactic Annotation Workspace

In the third workspace every segment of grouped handwriting strokes is assigned to a syntactic value, i.e. a symbol contained in the supported lexicon of the considered application domain. This is achieved by a) selecting the appropriate symbol entry from a structured list and b) performing a pen gesture to capture one or more segments from the sample being edited. The annotation process is also supported by color coding techniques and can be recontrolled by moving the pen on any annotated segment: The assigned syntactic value will then automatically be selected in the structured list mentioned above; additionally, all the segments assigned to the same entry will be highlighted. Fig. 3 gives an example of the syntactic annotation functionality. The acquired segmentation and syntactic annotation knowledge is postprocessed by our corresponding feature extraction methods and stored in a compressed database of reference patterns for every supported symbol class [2] [3].

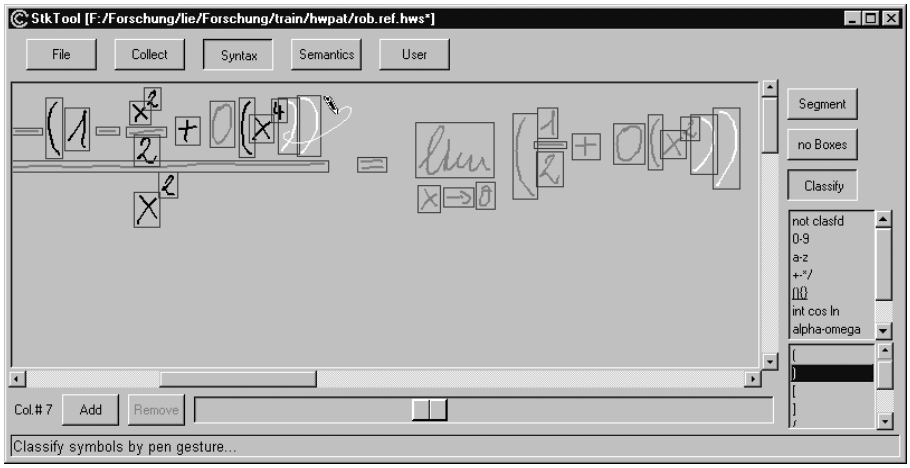

Figure 3: Syntactic Annotation Workspace. At first, one of the supported syntactic values (i.e. symbols) is selected – either by choosing it from the structured list on the right hand side, or by capturing a handwriting segment already assigned to that value (in this example: one of the white right braces on the right). Via pen gestures extra segments are then added on. Here, another two right braces are being assigned at a single blow. Different colors (here: dark grey or black, resp.) are used to distinguish between already or not yet annotated segments.

2.4 Semantic Annotation Workspace

Finally, the different syntactic constituents of a segmented and syntactically annotated sample must be allocated to semantic attributes. To this end, a so-called Semantic Structure, i.e. a compact hierarchical representation of the logical contents of a given handwritten expression, may be loaded into a separate window section inside the fourth workspace. In the special case of mathematical formulas, this object is automatically generated by the use of a conventional LaTeX compatible formula editor and subsequent data transformation due to text based chart parsing. The hierarchical structure of this representation is then - also by using intuitive pen gestures - mapped onto the collection of annotated symbols, so that successively any semantically closed subexpression of the given input (e.g. a fraction denominator) is correlated to a semantic unit of the corresponding semantic representation. Again color coding and (hierarchical) surrounding rectangle display are used to facilitate these steps and to quickly revise their consistency. Afterwards all the necessary (probabilistic) parameters which describe the semantic attributes contained in the choice and positional arrangement of the different symbols or symbol groups, respectively, are derived by the use of appropriate training procedures [4] [5]. The semantic annotation features are summarized in Fig. 4.

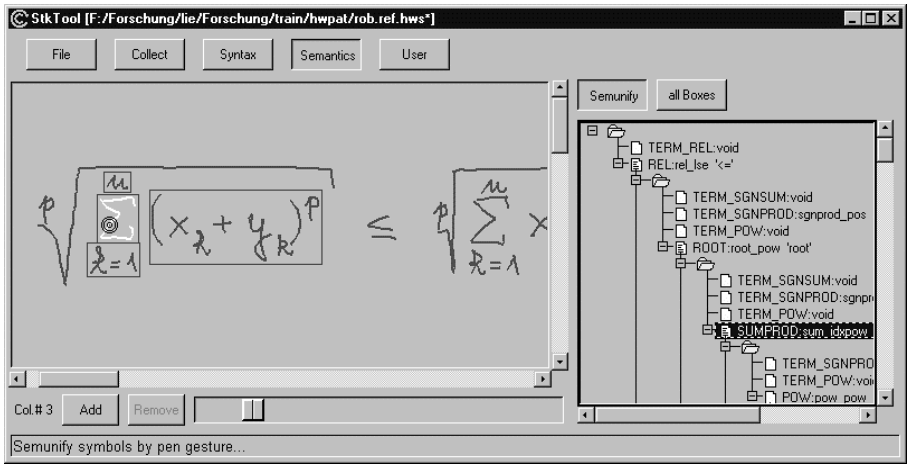

Figure 4: Semantic Annotation Workspace. In the right hand area the Semantic Structure of the given

sample formula is displayed as a hierarchical list. By selecting an entry the corresponding handwritten symbol(s) get highlighted (here: white), and bounding boxes appear around all semantic successor subexpressions.

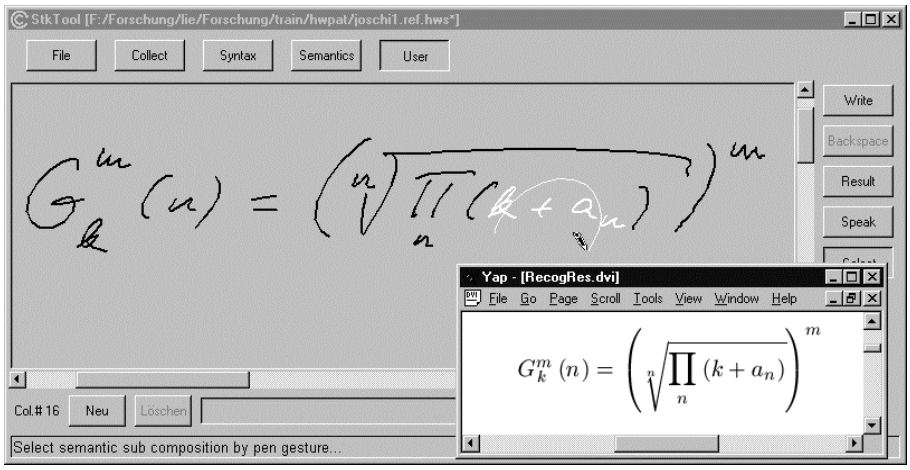

Figure 5: Interactive Formula Recognition. In this example, the given handwritten formula was correctly recognized and transformed to a typesetting format. By utilizing the recognized formula structure, the interface allows the user to perform "intelligent" pen gestures (i.e., only semantically self-contained subexpressions are selectable) in order to correct or modify formula parts.

3. RESULTS & CONCLUSIONS

We have demonstrated that intuitive pen gestures are well suited for a convenient and straightforward execution of segmentation and annotation tasks on handwritten or handdrawn input. By the use of our general pen-gestural interface the resulting sample data may systematically be evaluated on the different abstraction levels involving signal, syntactic, and semantic knowledge to supply a corresponding online recognition system with all the necessary probabilistic parameters. As an example, we implemented a robust probabilistic semantic decoder for handwritten mathematical formulas on the basis of the presented interface [5]. For interactive correction or modification tasks, our pen gesturing technique was also integrated in the end-user interface (cf. Fig. 5). In the future it should be examined to which extent the same approach succeeds for other application domains.

REFERENCES

[1] Welch B. B.: Practical Programming in Tcl and Tk**,** ISBN 0-13-022028-0, Prentice-Hall, New Jersey, USA, 1999.

[2] Intel Recognition Primitives Library Reference Manual, Order Number 637785-007, Intel Corporation, 1998.

[3] Hunsinger J. and Lang M.: *A Single-Stage Top-Down Probabilistic Approach towards Understanding Spoken and Handwritten Mathematical Formulas*, Proc. 6th International Conference on Spoken Language Processing (ICSLP 2000), Beijing, China, Vol. IV, pp. 386-389, October 2000.

[4] Hunsinger J. and Lang M.: *A Speech Understanding Module for a Multimodal Mathematical Formula Editor*, Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP) 2000, Istanbul, Turkey, Vol. IV, pp. 2413-2416, June 2000.

[5] Hunsinger J., Lieb R., Lang M.: *Real-time Structural Analysis of Handwritten Mathematical Formulas by Probabilistic Context-free Geometry Modelling*, to appear in: Proc. 2001 International Symposium on Intelligent Multimedia, Video & Speech Processing (ISIMP 2001), Hong Kong, China, May 2-4, 2001.