Interactive User Interface for Recognizing Online Handwritten Mathematical Expressions and Correcting Misrecognition

Vu Tran Minh Khuong, Khanh Minh Phan, Masaki Nakagawa
Department of Computer and Information Science
Tokyo University of Agriculture and Technology
Nakacho 2-24-16, Tokyo, Japan,
Email: s184468v@st.go.tuat.ac.jp, khanhpm@go.tuat.ac.jp

***Abstract*— This paper presents an interactive user interface for the recognition of online handwritten mathematical expressions (HMEs). Since the recognition results may have errors, our objective is to make a user interface by which the users could verify and edit the recognition result of an HME easily without rewriting it. Multiple editing gestures are proposed to correct recognition errors in symbol recognition, symbol segmentation, and structure analysis. We implemented prototypes and conducted a user study to compare two variations for the user interface and a few variations for gestures. The results show that participants prefer the user interface showing the recognition result of each symbol apart from its bounding box to the one which embeds the recognition result into its bounding box. Moreover, the user study confirmed that our correcting gestures enable the users to modify the recognition result as they expected.**

Keywords—User Interface, Handwritten Mathematical Expressions, Misrecognition Editing

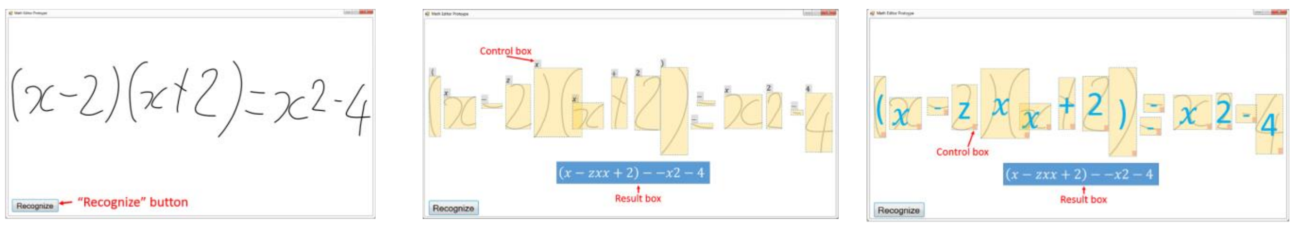
I. Introduction

Currently, there are three primary methods for entering mathematical expressions (MEs) into computers, by mathematical script languages, by mathematical formula editors and by computer recognition of handwritten mathematical expressions (HMEs). The mathematical script languages require the users to be familiar with a programming language (e.g., Latex) to input MEs. Meanwhile, the mathematical formula editor (e.g., MathType) often force the users to select predefined mathematical structure templates (e.g., fractions, superscripts, subscripts) from a menu or toolbar and then fill in the templates with numbers, letters, and so forth by typing on the keyboard. These two methods, however, have difficulties for students to use in the learning process, especially in answering descriptive questions.

On the other hand, the HME recognition-based method is a natural choice for inputting MEs. However, this method requires a highly reliable HMEs recognizer. HMEs are more challenging to recognize than natural languages since the structural ambiguity is higher. In fact, the recognizer of HMEs usually has problems with recognition errors. The errors could come from various levels of recognition: symbol segmentation, symbol recognition, and structure analysis. By providing a user interface with which the user can quickly correct recognition errors, the user experience could be significantly improved.

For handwritten Japanese, there are some similar works such as [1], [2]. For mathematics, Steve Smithies et al. proposed a prototype math editor that incorporated a set of simple procedures for correcting errors made in recognition [3]. In 2016, Taranta et al. introduced a dynamic system which used Math boxes for writing complicated mathematical expressions [4]. Besides, they incorporated some editing features which allowed the users to edit HMEs before and after their recognition and correct recognition errors. In the same year, Tokuda et al. also proposed an error correction user interface for inputting mathematical expressions by handwriting recognition [5].

In this paper, we prototype and evaluate two versions of the user interface for representing the HMEs recognition result. Moreover, we introduce various gestures for editing recognition errors. We also conduct a user study to evaluate and compare them to each other. In the rest of this paper, we will present user interface designs in section II. The editing

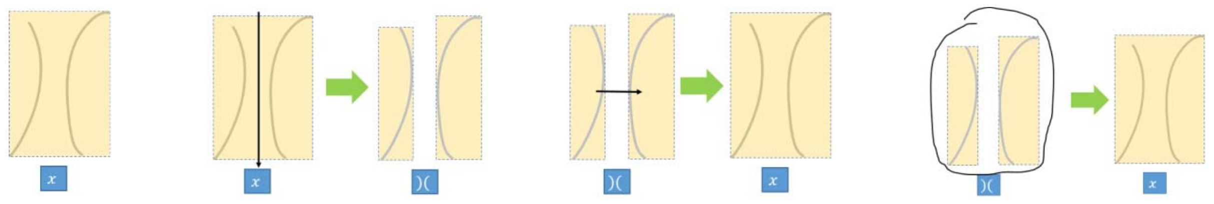


(a) Initial user interface.

(b) UI_on shows symbol recognition result on its bounding box.

(c) UI_inside shows symbol recognition result within its bounding box.

Figure 1. Two UIs for showing and editing HMEs recognition result.



(a) Original .

(b) Gesture for correcting an under-segmentation error.

(c) Connecting gesture for correcting an over-segmentation error.

(d) Encircling gesture for correcting an over-segmentation error.

Figure 2. Gestures for correcting symbol segmentation errors.

gestures are described in section III. The experiment results are presented and discussed in section IV. Conclusions are drawn in section V.

II. USER INTERFACE DESIGN FOR DISPLAYING RECOGNITION RESULT AND MISRECOGNITION CORRECTION

Usually, users do not know where misrecognitions come from. Hence, we aim to design a user interface which represents the HMEs recognition result such that the users perceive the recognition result and identify the recognition errors quickly. First, our user interface starts with an empty writing panel and a "Recognize" button. When the users complete their writing MEs, they can press the "Recognize" button to "tell" the system to recognize the entire HME (Fig. 1a).

After pressing the "Recognize" button, our user interface shows the recognition result in the result box below the HME and enables the users to edit recognition errors. Moreover, it shows the symbol segmentations using bounding boxes. To show the recognition result of each symbol, we consider two versions. The first version shows the symbol recognition result at the left position on its bounding box, as shown in Fig. 1 (b) (called UI_on). The second version embeds the symbol recognition result inside the bounding box, as shown in Fig. 1 (c) (UI_inside). The control box is associated with the bounding box, which captures some editing gestures (which will be introduced in section 3).

III. EDITING GESTURES

There are recognition errors in symbol segmentation, symbol recognition, and structure analysis. For each type of errors, we propose a couple of gestures for correcting recognition errors.

A. *Gestures for Editing Symbol Segmentation Errors*

The initial step in HME recognition is the symbol segmentation process. The wrongly segmented components are likely misrecognized in the following steps of HME recognition. Hence, we propose two types of gestures to correct segmentation errors when the user identifies missegmented components. In our user interface, the user can recognize segmented symbols using their bounding boxes.

The segmentation errors can be divided into two major types (the others can be a mixture of them):

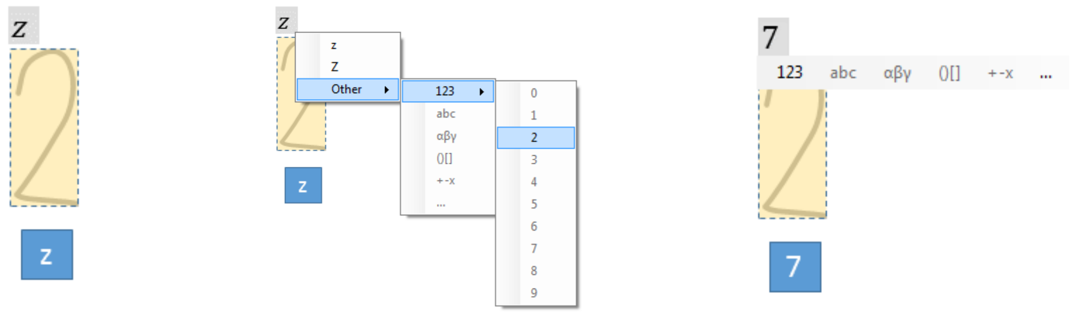
- Under-segmentation: A segmented component includes strokes belonging to multiple symbols.

- Over-segmentation: A segmented component includes a subset (but not the whole) of strokes of a single symbol.

The correction process of a segmentation error is triggered by the user to draw a gesture stroke on the missegmented segmentation components.

To correct an under-segmentation error, the user draws a stroke starting from outside the bounding box of the under-segmented component, penetrating it with separating the component and ending outside as shown in Fig. 2 (b).

On the other hand, for correcting an over-segmentation error, we propose two gestures: (i) a stroke starting from inside of the bounding box of an over-segmented component to inside of that of another as shown in Fig.2 (c); and (ii) a stroke encircling the bounding boxes of over-segmented



(a) Misrecognized symbol.

(b) Accessing symbol list by single-tap gesture.

(c) Accessing symbol list by multiple-tap gesture.

Figure 3. Symbol list for editing symbol mis-recognition by gestures.

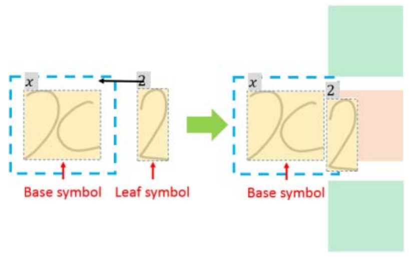


Figure 4. Effective region of base symbol "x" enclosed by the dashed blue rectangle

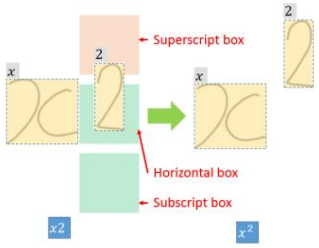


Figure 5. Recognition-based gesture for editing structure analysis

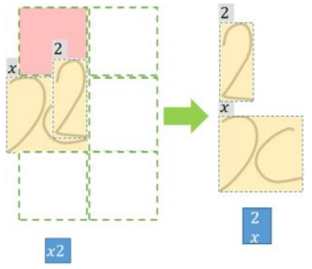


Figure 6. Non-recognition-based gesture for editing structure analysis

components as shown in Fig. 2 (d). For a symbol over-segmented into 3 or more than 3, the user may apply the first gesture successively or apply the second gesture. The first is suitable when strokes are crowded while the second allows the user to combine all the over-segmented components by only a single action.

The gestures (b) and (c) are effective for highly packed groups of symbols, while the gesture (d) is naïve and efficient for sparse symbols.

To give feedback to the user's gesture, the segmentation and symbol recognition result is updated immediately after the correction gesture is finished.

B. *Gestures for Editing Symbol Recognition Errors*

After segmenting symbols correctly, the symbol recognition process still may misrecognize the segmented components. Hence, we allow the user to correct the errors by interacting with the control boxes. We propose three approaches to handle this problem:

- Single-tap: In the first approach, a symbol list menu is shown by tapping the control box. The system displays a list of top N-candidates based on the recognition scores and the "Other" option. If the expected symbol is not in the candidate list, the user could access the full list of symbols by tapping the "Other" option followed by several groups and finally the expected symbol as shown in Fig. 3 (b).
- Multiple-tap: In the second approach, the users could switch among the recognition candidates by tapping the control box a few times. Then, tapping one more time shows the full list of symbols to allow the user to select the expected result, as shown in Fig. 3 (c).
- Multiple-and-long-tap: In the third approach, similar to the second approach, the recognition candidates are also switched by tapping the control box. However, the full list of symbols is accessed by a long tap gesture.

C. *Gestures for Editing Structure Analysis Errors*

This gesture is to correct misrecognitions in the structure analysis process. They occur at relations between two symbols. One symbol is the base of the relation, and the other is a leaf. For example, in the superscript relation of "x2", "x" is the base symbol while "2" is the leaf symbol. The structure analysis is reflected in positions and sizes of symbols in the recognition result. In order to trigger the process of correcting a structural relation error, the user taps on the control box of the leaf symbol and drag it into the effective region of the base symbol as shown in Fig. 4.

We propose two approaches: recognition-based gesture and non-recognition-based gesture. In the recognition-based gesture, after dragging the leaf symbol into the effective region of the base symbol, relational boxes are displayed such that each relational box corresponding to one of the following structural relations: horizontal, superscript, subscript, over, under, inside. The box corresponding to the current recognition result is highlighted in red, while the others are in green. The user drags the misrecognized symbol and drops it to the relational box corresponding to his/her expected structural relation. Based on the recognition result, the relational boxes are displayed according to the base symbol. If the base symbol is a fraction bar, the boxes of horizontal, over, under are displayed. If the base symbol is a root sign, the boxes of horizontal and inside are displayed. If the base symbol is a number or letter, the boxes of horizontal, superscript, and subscript are displayed as in Fig. 5. Hence, unexpected structural relations are excluded.

For the non-recognition-based gesture, the editing process is also triggered by dragging the leaf symbol to the base symbol's effective region. However, all the possible relational boxes are shown so that the user could arbitrarily modify the expressions as shown in Fig. 6. Particularly, the user could drag the symbol he/she wants to modify and drop it to the desired box. Again, the system gives feedback by highlighting the selected box in red. After recognizing the gesture, the system will translate the leaf symbol and update the recognition result.

IV. EXPERIMENTS

Our experiment is divided into two steps: the first step is focusing on evaluating and comparing the effectiveness of the two user interface prototypes and the editing gestures while the second step is evaluating the editing experience when using the user interface prototype selected.

In the first step, we let the users try the two versions of the user interface and the editing gestures at the same time without any instruction or training. For the two versions of the user interface, we presented them at the same time and let the participants evaluate. For evaluating the editing gestures, we gave sample expressions which contained recognition errors and asked participants to correct them. After the users' try, we

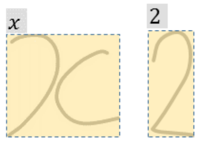


Figure 7. Sample pattern for evaluating a gesture to edit structural relation.

asked them to give their evaluation on the gestures even if they succeed or not. Most of the participants could imagine the correct gestures without explanation. Then, we explained the correct gestures when the user could not figure out how to perform the gestures. In the second step, we let the users try correcting misrecognition of the HME in Figure 1 (a) on the user interface prototype selected based on their evaluation in the first step. The editing gestures are also selected according to their evaluation.

We recruited 20 participants. Their age ranged from 21 to 30 (25.7±2.66). Fifteen participants were male and five participants were female. Eighteen participants were right-handed and two are left handed. Five participants had never use a pen or touch-based devices for writing, note-taking, and so on, whereas the others had some prior experience with the devices. The participants' backgrounds included information technology (16), construction (1), chemistry (1), biology (1), and mechanical engineering (1). This experiment was conducted on Surface pro3: Intel Core i5-4300U CPU (2.5 GHz) RAM 4Gb, Microsoft Windows 10 Professional.

A. Comparing two interface versions and editing gestures

In order to compare the two versions as well as compare the editing gestures, we employed the 5-point Likert Scale from 1 (strongly dissatisfied with the user interface or editing gesture) to 5 (strongly satisfied with the user interface or editing gesture). For each case, we collected the users' evaluation and obtained the average Likert points. First, we presented the two versions of the user interface, as shown in Figure 1 (b) and Figure 1 (c) on the same screen and asked the participants to identify recognition errors. We asked whether the proposed user interface prototypes enabled them to perceive the recognition result and identify recognition errors easily. UI_on achieved 3.85 points, whereas UI_inside achieved 3.65 points. One of the main reasons UI_inside was evaluated lower is due to the representation of the symbol recognition results. When the patterns are written densely, their bounding boxes tend to overlap to other ones, and so does the symbol recognition results. Moreover, the representation visually confuses the baseline of the expression because of the large character size. On the other hand, UI_on shows the symbol recognition results apart from the handwritten patterns with small character size. Hence, it does not affect the handwritten pattern and its baseline as much as UI_inside does. However, we still need to consider the size of recognized symbols for both the versions.

For evaluating the gestures to correct symbol segmentation errors, we showed two handwritten samples as shown in Fig. 2 (a), and asked the participants to use the gesture to separate the strokes to form ")(" pattern or to merge them to form the "x" pattern and evaluate each gesture. The users were supposed to perform the gesture in Fig. 2 (b) for separating the strokes on both samples and perform the gestures described in Fig. 2 (c) for merging one sample and the gestures described Fig. 2 (d) for merging the other. According to the participants' evaluation, the connecting gesture is clearly preferred with 3.95 points (encircling gesture: 2.30). By applying paired samples t-test, we found that the connecting gesture was significantly evaluated higher than the encircling gesture (p-value $< 3.1\times10\text{-}5 < 0.05$). While the connecting gesture only requires the gesture stroke touch the over-segmented components, the encircling gesture requires the gesture stroke to encircle both the bounding boxes of over-segmented components.

For evaluating the gestures to correct symbol recognition errors, we showed three misrecognized handwritten samples, as shown in Fig. 3 (a). Then, we asked the users to change all the symbol recognition to "2". The users were supposed to figure out three different gestures described in Section III B to correct the misrecognition and evaluate them. According to the participants' evaluation, the single-tap gesture was dominant to the others with 3.70 points (multiple-tap gesture: 2.60, multiple-and-long-tap gesture: 2.70). We conducted two paired samples t-test: one with the hypothesis that the single-tap gesture was evaluated higher than the multiple-tap gesture (the p-values was less than $5.2\times10\text{-}3 < 0.05$), one with the hypothesis that the single-tap gesture was evaluated higher than the multiple-and-long-tap gesture (the p-values was less than $0.02 < 0.05$). Therefore, we concluded that the single-tap gesture was significantly evaluated highest. The reason is that the single-tap gesture could show multiple recognition candidates quickly by just one tap (and shows the symbol list as a context menu at the same time) while the others require multiple taps to switch to the next candidate and to access to the symbol list. However, some participants could not find a way to correct the misrecognition in this task. The reason is that they tried to rewrite the misrecognized patterns again instead of interacting with the control boxes, which is not allowed in our prototypes. Hence, adding the rewriting function inside the bounding box would be necessary.

For evaluating the gestures to edit the structural relation, we showed two handwritten samples, as shown in Fig. 7. Then we asked them to edit the relation between symbols to "x^2" or "x_2". The user were supposed to figure out the gestures described in Section III C. According to the participants' evaluation, the recognition-based gesture achieved an average of 3.60 points while the non-recognition-based gesture achieved that of 3.20 point. The recognition-based gesture is probably preferred since it could eliminate the inappropriate relational boxes and let the participants concentrate on the correcting process. However, there are some comments about the difficulty of interacting with the small control boxes. Enlarging the interacting area is an idea for improvement.

B. *Evaluating user experience*

After letting the participants experience the gestures, we let the user try correcting the expression in Fig. 1 (a) on the user interface Fig. 1 (b) or Fig. 1 (c) based on their evaluation in the first experiment. We asked the users to edit the recognition result to “$(x-2)(x+2)=x^2-4$”. Hence, the expression contains symbol recognition error (the first “2”), under-segmentation error (the “)(”), over-segmentation error (two “-” of the “=”) and structural error (“x” and “2” on the right-hand side of “=”). Then, we asked a series of questions to obtain the participants’ evaluation of the prototype user interface. As shown in Table I, the majority of users agree or strongly agree that the user interface enables them to correct the HMEs misrecognition as they expected.

TABLE I. HOW DO YOU RANK THE EFFECTIVENESS (CAN YOU EDIT RECOGNITION ERRORS AS YOU LIKE)?

Extremely likely	5 (25%)
Likely	10 (50%)
Neither	4 (20%)
Unlikely	0 (00%)
Extremely unlikely	1 (05%)

Additionally, the majority of users agree or strongly agree that the user interface and the gestures are efficient, as shown in Table II.

TABLE II. HOW EFFICIENTLY CAN YOU EDIT MATH EXPRESSIONS (HOW QUICKLY, EASILY, AND SO ON)?

Extremely efficient	4 (20%)
Efficient	10 (50%)
Neither	0 (00%)
Inefficient	5 (25%)
Extremely inefficient	1 (05%)

They also agree or strongly agree that the interactions with the user interface are clear and understandable, as shown in Table III. Besides, the majority of participants say that it is easy to learn, as shown in Table IV. However, they thought it would be easier if there are instructions beforehand in general.

TABLE III. DO YOU FEEL THAT YOUR INTERACTION WITH THE USER INTERFACE IS CLEAR AND UNDERSTANDABLE?

Extremely likely	3 (15%)
Likely	10 (50%)
Neither	4 (20%)
Unlikely	3 (15%)
Extremely unlikely	0 (00%)

TABLE IV. DO YOU FEEL IT IS EASY TO LEARN TO USE THE USER INTERFACE?

Extremely likely	4 (20%)
Likely	10 (50%)
Neither	4 (20%)
Unlikely	2 (10%)
Extremely unlikely	0 (00%)

Overall, the majority of participants are satisfied with the user interface as shown in Table V, and they preferred the designed user interface to the traditional math input user interface such as math editor and math language as shown in Table VI.

TABLE V. HOW ARE YOU SATISFIED WITH THE USER INTERFACE?

Extremely satisfied	2 (10%)
Satisfied	11 (55%)
Neutral	6 (30%)
Dissatisfied	1 (05%)
Extremely dissatisfied	0 (00%)

TABLE VI. WHICH DO YOU PREFER AMONG THE RECOGNIZE/EDIT MATH INPUT, MATH EDITOR, AND MATH LANGUAGE FOR INPUTTING MATH FORMULA ON THE COMPUTER?

The recognize/edit math input	13 (65%)
Math editor	4 (20%)
Math language	3 (15%)

V. CONCLUSION

In this paper, we have presented a design of the user interface for correcting recognition errors of handwritten math expressions. According to the user study, the majority of participants agree that our design of the user interface allows them to edit the result of HME recognition as they want. Also, from the user study, we obtained several ideas for improving the user interface.

The flowchart of the editing process is as follow.

- If the user taps on the control box, the system enables the user to modify symbol recognition.
- If the user drags the control box, the system enables the user to modify the structure relation.
- If the user draws a gesture stroke, the system handles the editing segmentation.
- Otherwise, the system discards the stroke.

ACKNOWLEDGMENT

This work is being partially supported by the Grant-in-Aid for Scientific Research (A)-19H01117.

REFERENCES

[1] H. Bandoh, T. Fukushima, N. Kato, and M. Nakagawa, "User Interfaces for Correcting Errors in Writing-box-free Recognition of Handwritten Text (in Japanese)," Trans. Of IPS Japan, Vol. 43, No. 6, pp. 1996-2005, 2002.

[2] M. Nakagawa and M. Yorifuji, "Error Correction Gestures for Free-format Text Input by Pen Interface (in Japanese)," Proceedings of the Human Interface Symposium, pp. 43-46, 2005.

[3] S. Smithies, K. Novins, and J. Arvo, "A handwriting-based equation editor," Proceedings of the 1999 conference on Graphics Interface, pp. 84-91, 1999.

[4] E. M. Taranta II, A. N. Vargas, S. P. Compton, and J. J. Laviola Jr., "A Dynamic Pen-Based Interface for Writing and Editing Complex Mathematical Expressions With Math Boxes," ACM Transactions on Interactive Intelligent Systems (TiiS), Vol. 6, No. 2, Article no. 13, 2016.

[5] S. Tokuda, A. D. Le and M. Nakagawa, "Design and Prototyping of Error Correction Interface for Inputting Mathematical Expressions by Handwriting Recognition (in Japanese)," IPSJ SIG Technical Report, Vol. 2016-CE-133, No. 8, pp. 1-6, 2016.