Error Repair in Human Handwriting - An Intelligent User Interface for Automatic On-Line Handwriting Recognition

Wolfgang Hürst, Jie Yang, and Alex Waibel

Interactive System Laboratories
Carnegie Mellon University, Pittsburgh PA, U.S.A.
University of Karlsruhe, Germany
Email: {wolfgang+, yang+, ahw}@cs.cmu.edu

Abstract

Several important factors, such as recognition accuracy, user acceptance, and system usability, have to be considered in designing an interface of a handwriting recognition system. Since both users and recognition algorithms make mistakes, it is desirable for the user interface of a handwriting recognition system to have mechanisms recovering from errors. In this paper we address the problem of error repair in on-line handwriting recognition. First, we perform a user study on error occurrences and corresponding repair patterns in human handwriting. Based on a data analysis, we have identified typical types of errors and repair patterns. We then propose methods to deal with error repair in an on-line handwriting system. We have developed a prototype system to demonstrate and evaluate the proposed error handling mechanisms. The system extends NPen^{++}*, an on-line handwriting recognition system developed in our lab, by providing error repair abilities to users in addition to its high recognition rate. The experimental results indicate that the error handling mechanisms can significantly improve the system performance in case of the data containing error repair.*

1. Introduction

Automatic handwriting recognition provides an important channel for human-computer communication. Although intensive research has been directed to improve handwriting recognition technology, recognition results remain inherently unreliable. User studies showed that even humans are not able to achieve a recognition accuracy of 100% [1]. It is very unlikely that we will ever be able to develop an error-free recognition engine. Furthermore, our studies show that recognition errors result from not only the system but also human mistakes, e.g., misspelled words. Therefore, even with a "perfect" recognition engine we would still face the problem of misinterpretation. Moreover, users demand for a comfortable interface with high usability (in [2] a study can be found which analyses the relationship between user satisfaction and recognition performance of pen-based interfaces). Research in speech recognition [3] has demonstrated that even with unreliable baseline spoken language interpretation technology it is possible to significantly reduce the time to interact with a system via spoken language by various error repair strategies. Therefore we should not wait for a "perfect" system before we use handwriting recognition technologies in human-computer interaction but concentrate on the design of useful and usable interfaces which have the ability to perform repair and which are able to recover from occurring errors.

In this paper, we investigate the problem of error repair in an on-line handwriting recognition system. The objective is to develop concepts that could minimize the user's efforts to recover from errors in an on-line handwriting interface. While most current pen-based systems only offer repair possibilities by providing additional buttons (e.g., a "clear"-button) or by recognizing a fixed set of gestures, we aim at providing users with a handwriting interface that can recover recognition errors in natural and flexible ways. We present a user study to show possible errors and repair patterns in pen-based handwritten input and introduce a general framework for repair handling. In order to demonstrate the feasibility of the proposed concepts, we developed a prototype system, that consists of a user friendly interface, repair handling mechanisms, and the NPen^{++} recognition engine, which is a system for high accuracy writer independent on-line handwriting recognition of continuously written single words (see [4, 5] for more information on the NPen^{++} recognition tool). Our prototype system can be used for user study and investigating error repair strategies. We conclude with an evaluation of the proposed methods and discuss future research areas of error repair in our on-line handwriting recognition system.

2. The Problem of Error Repair

2.1. Errors in Human Handwriting

Before we discuss different methods to avoid errors or recover from errors in an on-line handwriting recognition system, we should investigate what kind of errors usually leads to a wrong recognition result. Based on study from literature [1], errors in human handwriting of textual material include:

- discrete noise events,
- badly formed shapes,
- input that is legible by humans but not by the algorithm (see next section),
- badly spelled words,
- words that are unsolicited in the data collection process,
- canceled material,
- device generated errors.

It is obvious that some of these errors could be avoided and some of them could be recovered by adding repair handling mechanisms to the user interface.

2.2. Errors in On-line Handwriting Recognition

A handwritten input which is legible to humans may not be recognized by a recognition engine. We discuss this problem based on the NPen^{++} recognition engine.

In on-line recognition the input contains not only spatial information (i.e., coordinates) but also time information. In the NPen^{++} system there exists some information about the x and y coordinates of the handwritten signal and about pen-down and pen-up events. Therefore we have an input signal S which is a time ordered sequence of coordinates: $S = ((x_0, y_0, p_0), (x_1, y_1, p_1), (x_2, y_2, p_2), \ldots, (x_n, y_n, p_n))$, where p_i is "0″ when a pen-up event occurred and "1″ else. Based on this signal a preprocessing step is done which does a normalization of the signal S to remove undesired and disturbing variability and calculates some feature vectors. This results in a time ordered sequence of feature vectors each representing some information useful in the recognition process. The recognition is based on a fixed set of words, i.e., the dictionary. Each word is composed of its letters. Each letter in turn is composed of three states. In the first step a Time-Delay Neural Network (TDNN) calculates hypotheses of each state for every letter from the alphabet. Based on the word models from the dictionary the Viterbi search is performed on the output of the Neural Network to find the best word hypothesis. An illustration of the whole recognition process is shown in Figure 1. A detailed description of the NPen^{++} system can be found in [4, 5].

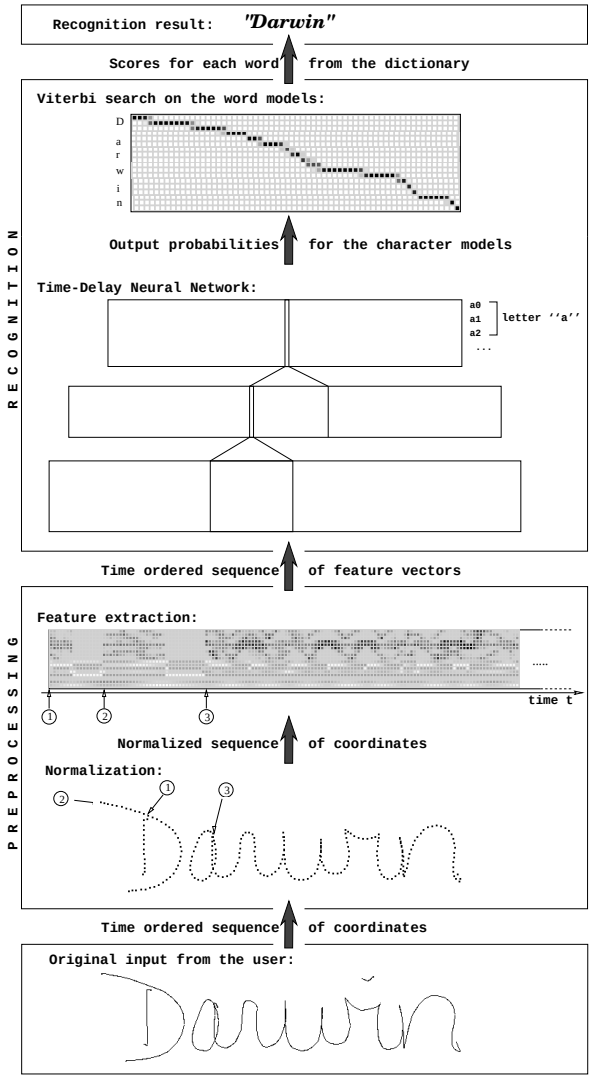


Figure 1. The recognition process of the NPen^{++} system.

The mapping from the TDNN output units to the word models from the dictionary assumes that all the coordinates belonging to one letter are written within a connected time interval. This, however, is not always true in human handwriting. Typical examples are delayed t-strokes and i- or j-dots. Hence a recognition result can be wrong, because the algorithm is not able to match the time sequence of input features correctly to the word models. This is sometimes called the "delayed-stroke problem".

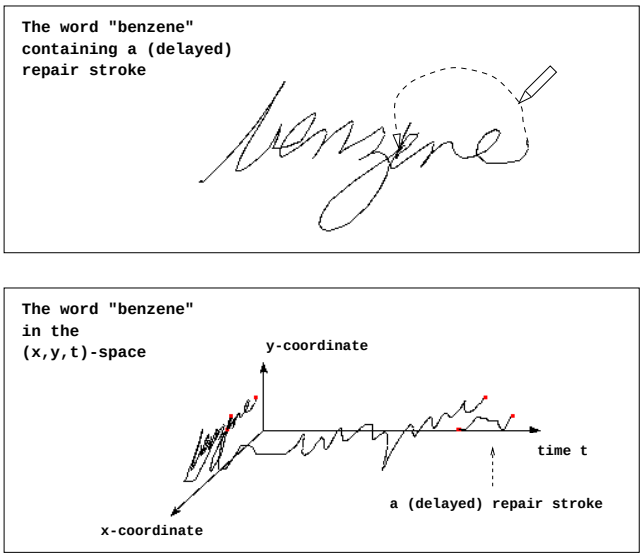


Figure 2. Example for a "Completion".

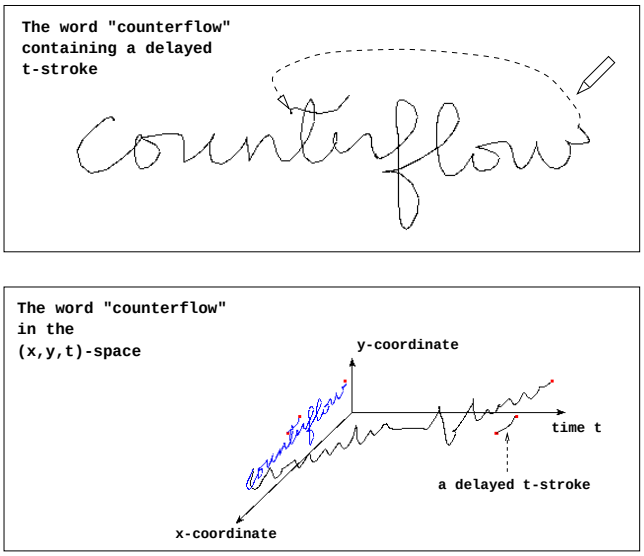


Figure 3. Example for a delayed t-stroke.

Several approaches exist to deal with this problem. For example, strokes can be reordered according to their x-coordinate values in the preprocessing step [6]. The NPen^{++} System handles delayed strokes at the preprocessing layer by introducing a so called "hat-feature". With a heuristic the system detects delayed strokes and dots and removes them from the sequence of coordinates. The hat-feature of a remaining feature vector will be set to "1" if its corresponding x-value lies in the same area as the removed coordinates otherwise it will be set to "0". An example of this can be found in Figure 1 where the i-dot from the word "Darwin" does not appear in the normalized sequence of coordinates after the first preprocessing step. The normalization removes the delayed i-dot. In the feature extraction step the hat-feature of the remaining feature vectors from the letter "i" are set to "1".

In this paper, we propose a unified framework to handle repair and delayed strokes together. In fact, in an on-line handwriting recognition system such as NPen^{++}, delayed t-strokes and i- or j-dots can be considered as a repair. We illustrate this by the following example. Figure 2 shows a handwritten word that contains a repair. After writing the word "benzene" the user corrected the error "badly formed shape" by adding some coordinates to the second "e" (we will refer to this type of repair as a "completion"). The correction is difficult to notice by only looking at the bitmap. It is easier to detect the delayed repair stroke from (x, y, t) space shown in the bottom of Figure 2, which indicates the time of downwriting. A sample that contains a delayed t-stroke is shown in Figure 3. By looking at the trajectory in (x, y, t) space, the connection between repair and the delayed-stroke problem in on-line handwriting recognition can be easily observed. In both examples, the delayed strokes may cause recognition errors. Therefore the delayed-stroke problem can be seen as a special kind of repair and should be handled under the same global framework.

2.3. Repair in On-line Handwriting Recognition

In order to learn more about repair types and patterns occurring in on-line handwriting recognition we performed an empirical study on human handwriting. The database we used consists of 3466 single words and 3410 text sequences each containing about eight words. The data was collected with a digitizing tablet and without any feedback from a recognizer. The users were asked to write the words that appeared on a computer screen. The data might have some bias because the users were not asked to do any corrections during the data collecting process. Users might act in different ways if they were allowed to repair errors. But this database is useful to analyze typical repair patterns and features in human handwriting.

Based on the list of typical errors in human handwriting introduced in Section 2.1, we analyzed the database. We found that about 13% of the words and 23% of the word sequences contained errors. We further investigated these wrong words and checked if any of them contained a repair. We discovered that users tried to do corrections sometimes even they were not asked to do any correction. This confirms our proclaim that users demand for repair and error handling features.

By analyzing the different repair styles in the database we

defined the following types of repair for on-line handwriting recognition:

- *Deletions* (see examples in Figure 4)
- *Overwriting* (see examples in Figure 5)
- *Completions and Insertions* (see examples in Figure 6)

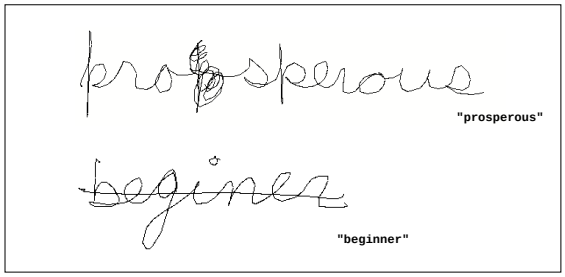


Figure 4. Examples for repair type "Deletion".

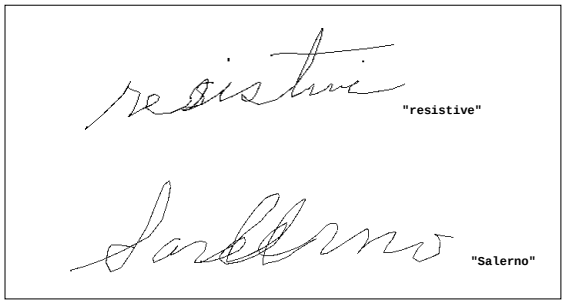


Figure 5. Examples for repair type "Overwriting".

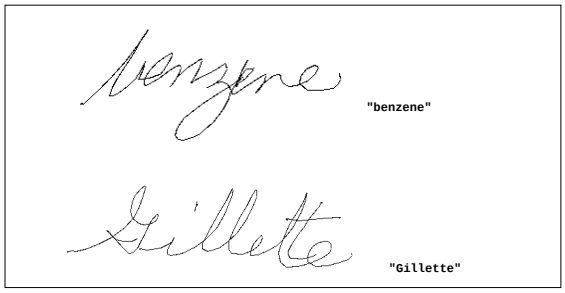


Figure 6. Examples for repair type "Completion" (top) and "Insertion" (bottom).

In this paper we only consider insertions within a word, e.g., entering an letter between two others. We exclude insertions which are written over or under the word and combined with gestures used to indicate the position of an insertion. In fact, these excluded insertions represent a very small percentage in our database.

3. Repair Handling

In order to handle error repair at the preprocessing level, we can use the same framework to handle error repair and delayed strokes, like already discussed in section 2.2. The input trajectory $S = ((x_0, y_0, p_0), (x_1, y_1, p_1), (x_2, y_2, p_2), \ldots, (x_n, y_n, p_n))$ containing a repair will be transformed into a "clean" trajectory $S^* = ((x^*_0, y^*_0, p^*_0), (x^*_1, y^*_1, p^*_1), (x^*_2, y^*_2, p^*_2), \ldots, (x^*_m, y^*_m, p^*_m))$ which can be handled by a general recognition engine. In S^* all the coordinates belonging to one letter are connected with each other in an ordered sequence, deleted or overwritten parts of the handwriting will be removed from the input trajectory.

However, identifying the repair mode of performing the required repair is much more difficult than that of handling the delayed-strokes. We cannot use simple heuristics like those used for the delayed strokes. For example, it is impossible to determine if a delayed stroke is a t-stroke or a repair gesture that crosses out the letter "l" without the use of context information and feedback from the recognition engine. Since there is a great variation in repair patterns, it is very difficult for a recognition engine to handle all the repair patterns. In other words, a general recognition engine cannot take care of all these variations without reorganizing the complete structure of the system and recognition algorithms.

In order to solve this problem, we propose to handle error repair at the preprocessing level. In fact, by interacting with the user, the system can handle repair at the preprocessing level without any feedback from the recognizer. In this way we can make minimal modification to the current recognition engine and benefit from interacting with users. The system can use some simple heuristics to detect, classify, and handle error repairs although there is no guarantee that such an effort will always be successful. The heuristics will allow a fast implementation and therefore fast indication of the result of the repair handling algorithms. Therefore we can offer an interactive way to a user to recover from errors.

Our goal is to give users more freedom to use different repair strategies other than to force them to use certain special gestures. Consider the following scenario: a user has crossed out the letter "l" in a word by making a horizontal stroke that is interpreted from the system as a t-stroke. If the user sees that no deletion happens, he/she will continue to delete the letter until it disappears. We could take advantage of this repair pattern in an on-line handwriting interface. The basic idea is to detect this "repeated delete" pattern. For example, a horizontal stroke which is made three times

is more likely to be a deletion than a single stroke. The heuristics we propose for repair classification are designed in a way that they are able to detect this kind of "repeated" repair. By interacting with the user, the system can minimize the efforts to recover from errors.

To transform the trajectory S containing a correction to the "repaired" trajectory S^*, the system has to decide which coordinates should be removed from the trajectory S and which should be reinserted to another position. In case of a deletion both, the repair signal and the repaired parts of the word, have to be removed from the original input. If an overwriting happened only the overwritten part of the word has to be removed from the sequence of coordinates. The strokes performing the repair should be inserted at the correct position. In case of a "pure" insertion or completion, no part of the word has to be deleted. Only an insertion of the repair signal has to be done. We do these operations on the basis of "up-down strokes", i.e., whole groups of coordinates in an interval between a local maximum and the following local minimum (or vice versa). An example for the up-down strokes of a given handwriting can be found in Figure 7. Our experience has showed that this method works very robustly.

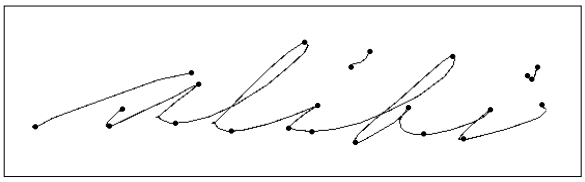


Figure 7. Example for the up-down strokes of the word "alibi".

A common pattern of repair in our database was that the writers violated the rule of writing from left to right. We use this characteristics to separate all the up-down strokes from the sequence S into repair and none-repair strokes. For each up-down stroke we calculate some features and compare them with the features of the previous ones. If the features of the new up-down stroke do not fit into the sequence of features of the preceding up-down strokes, it will be classified as a repair. For example, the i-th up-down stroke is seen as a repair, if

$$x_i^{min} < \Phi \cdot min\{x_{i-1}^{min}, \ldots, x_{i-5}^{min}\}$$

where x_i^{min} is the smallest x-value of the i-th up-down stroke in S and Φ is a fixed threshold. As a result of this calculation we get a set of up-down strokes classified as a repair and a set of "repaired" up-down strokes. These are the parts of the word which have been overwritten by the repair signal.

The procedure of classification and repair handling such as *Deletion, Overwriting,* and *Completion/Insertion*, is illustrated by the diagram shown in Figure 8.

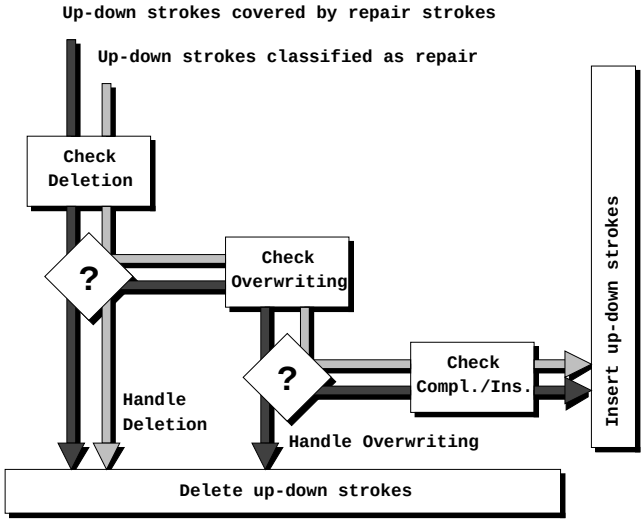


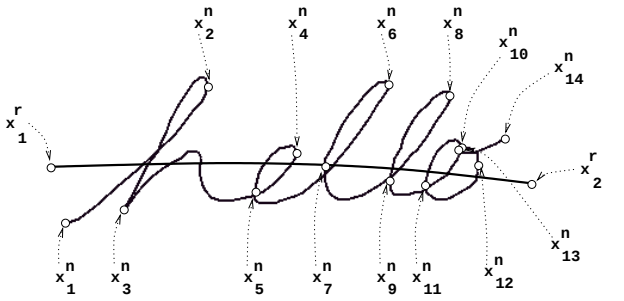
Figure 8. The overall repair handling schedule.

First the system detects deletions. It uses some heuristics to compare the up-down strokes that have been classified as repair strokes with the rest of the strokes. If there exist big differences between the attributes of these two sets, a deletion is classified. The heuristics compare the two sets of coordinates based on their size and extension in x- and y-directions. An example for one of these heuristics can be found in Figure 9. Here the following heuristic discovers a repair:

$$\max_{j>1} |x_j^r - x_{j-1}^r| > \Psi \cdot \max_{i>1} |x_i^n - x_{i-1}^n|$$

where Ψ is a fixed threshold, x_i^n is the i-th element of the sequence of the x-values from the coordinates that indicate a change in the writing direction from left to right (and vice versa) in the part of the handwriting classified as none-repair, and x_j^r is the j-th element of the corresponding sequence of repair coordinates.

This heuristic will ignore a relatively small repair stroke. Due to the interactive indication of the repair handling, the user will realize this misclassification and try to repair again. With this heuristic, we can add another heuristic: if a stroke which has been classified as a repair contains much more coordinates than a normal number of coordinates for such a stroke, the stroke is classified as a deletion, i.e., if R is the set of coordinates classified as a repair and $\bar{R}$ is the set of coordinates that lie in about the same area as the ones in R,



$$|x^r_1 - x^r_2| > \Psi\,|x^n_3 - x^n_4| = \Psi \max_{1<i<15}\{|x^n_i - x^n_{i-1}|\}$$

Figure 9. Example for one heuristic used to classify the repair type "Deletion".

then R is a deletion, if

$$number\ of\ coords\ in\ R > \Omega \cdot number\ of\ coords\ in\ \bar{R}$$

where Ω is a fixed threshold. This heuristic is most likely to apply if a user repeats his/her repair because of a misclassification in the first place.

With the heuristics we introduced in our systems for deletion handling we cover the two main types of a deletion found in the database we introduced in section 2.3:

- a few, but compared to the "normal" handwriting relative long strokes (like, for example, in the bottom of Figure 4),
- many (sometimes very short) strokes placed in an unpredictable order in a specific area (like, for example, in the top of Figure 4).

If no deletion is classified, i.e., if none of the heuristics detects a high dissimilarity between the repair strokes and the rest of the handwriting according to some writer dependent attributes of the handwriting style, the repair strokes will be considered as overwriting strokes. We have also specified some heuristics to detect and remove overwritten parts of a word from the input trajectory. The heuristics compare the bounding boxes of the repair strokes and the bounding box of each up-down stroke from the remaining input trajectory. If a high overlap between these two areas occurs an overwrite is classified.

After classifying the parts of the written word that have to be removed due to overwriting, the repair strokes, i.e., the coordinates performing the overwrite, have to be inserted into the remaining trajectory. This situation is similar to the case of completion or "pure" insertion. Therefore these two are considered together in the last step of our repair handling schedule (Figure 8). To find the best position to insert the repair strokes into a sequence of coordinates, we defined a distance measure that calculates a score for every possible position. We restricted these positions to the local maxima and minima, i.e., to the borders of the up-down strokes. The repair trajectory is inserted at the position with the lowest distance measure.

After the use of these heuristics, the parts of the trajectory which were classified as a deletion or as the overwritten part of the word have to be removed from the input trajectory. The ones classified as overwriting, completion, and insertion must be inserted at a position based on the previous heuristic. At the end of the repair, the repaired trajectory has to be smoothed to fill the gap originated in cause of a deletion.

With this repair handling scheme a misclassification of a "regular" handwriting as a repair, i.e., a misclassification at the very first check, will cause no harm, because the strokes which were wrongly classified as a repair are just "reinserted" at the correct position.

4. System Evaluation

We performed several experiments to evaluate the feasibility of the proposed methods. Since the database we used for user study was collected without requesting users to do any repair, the database is not suitable for us to evaluate repair handling algorithms. It is not an easy task to collect data to evaluate repair handling systems because any hint to a user can influence his/her behavior. In our case the evaluation is even more difficult because our heuristics are designed in a way that they handle repair directly by interacting with the users.

We defined a set of 200 words consisting of equally sized groups which contained the following errors: additional letters, wrong letters, permuted letters, and missing letters. The resulting word pairs of correct and wrong words were chosen from the dictionary in a way that both of them make sense, or they look similar. For example, "hair" and "chair" are a word pair with one (wrong) additional letter, and "yosemite" and "yosenite" look similar. To get statistically reliable results we chose these word pairs randomly from the dictionary. For example, by randomly selecting the word "hair" and searching for another word that only differs by one additional letter we found the "hair"-"chair" word pair.

With these 200 word pairs we did a data collection with four users. Each of them was asked to write a word, for example, to write "chair". After finishing the input the user was asked to do repair, for example, to repair "chair" to "hair". We collected data twice. First, we did not show any feedback from the recognizer or the repair handling tool to the user so that we could observe how users reacted when

without error handling	9%
with error handling	42%
with interactive error handling	73%

Table 1. The normalized evaluation results

they were asked to do some repair. We discovered that the repairs that the users have made are very similar to those in the database we analyzed in the previous section. Then we showed the wrong written words to the users again and asked them to repair these words second time. We used our repair tool to directly indicate the resulting repair action. In this way we got some new data that can be used for testing the proposed approach.

We evaluated the data containing repair with the NPen^{++} recognition engine (see Figure 1). The dictionary used for evaluation is a subset of the Wall Street Journal Dictionary, containing about 50,000 words. The recognition accuracy of the system for "clean" data is 88%. With this baseline system, we have studied three cases: (1) without any repair handling; (2) with repair handling but without repair indication to the user; and (3) with repair handling and with repair indication to the user. The experimental results showed that 8% of the words were recognized correctly without any repair handling. This poor performance is expected because the NPen^{++} system does not contain any repair handling features. By using our heuristics but without repair indication to the user, the recognition accuracy was 37%. By using our heuristics and interacting with the user, the recognition accuracy increased to 65%. We can normalize these results with the baseline system (88%). The normalized results are shown in Table 1. These results indicate and confirm our proclaim that repair handling can be done even without any feedback from the recognition engine by integrating the user into the repair handling process.

5. Conclusion and Future Work

In this paper we have studied the problem of error repair for on-line handwriting recognition. By investigating user repair patterns and recognition errors caused by repair in current systems, we have proposed repair methods to allow a user to recover from errors in a natural and easy way. The basic idea is to interactively handle the error repair at the preprocessing level. In our approach, the normal recognition and error repair are handled under the same framework. For example, delayed strokes are considered as a special kind of repair. We have introduced heuristics to detect which mode the user is in and give immediately feedback to the user during the repair process. We have developed a prototype system to demonstrate the proposed methods. The initial experiments have shown some promising results.

The system also provides a powerful tool for studying the problem of error repair in on-line handwriting. This is one of our current research focuses. An interesting problem is how the user behavior changes if feedback exists from not only the preprocessing level but also the recognition engine. In addition, we are working on improving usability of the system by extending our prototype to cover a greater variety of repair patterns. Moreover, we will try to combine the current methods with some well known gestures to cover more error repair patterns. We already implemented some methods that allow to repair not only the handwritten trajectory but also the recognition result. This problem is easier than repair within a handwritten signal, because repair and repaired signal are well separated, i.e., no detection of the repair mode has to be done. We will further evaluate these methods with extensive user studies.

References

[1] L. R. B. Schomaker, “User-Interface Aspects in Recognizing Connected-Cursive Handwriting”, *Proceedings of the IEE Colloquium on Handwriting and Pen-based input*, 1994.

[2] Clive Frankish, Richard Hull, and Pam Morgan, “Recognition Accuracy and User Acceptance of Pen Interfaces”, *CHI’95 Conference Proceedings*, 1995.

[3] Bernhard Suhm, Brad Myers, and Alex Waibel, “Interactive Recovery from Speech Recognition Errors in Speech User Interfaces”, *Proceedings of the ICSLP 96*, 1996

[4] S. Manke, M. Finke, and A. Waibel, “NPen^{++}: A writer independent, large vocabulary on-line cursive handwriting recognition system”, *Proceedings of the International Conference on Document Analysis and Recognition*, 1995.

[5] S. Manke, M. Finke, and A. Waibel, “The Use of Dynamic Writing Information in a Connectionist On-Line Cursive Handwriting Recognition System”, *Advances in Neural Information Processing 7*, 1994.

[6] Robert H. Kassel, “A Comparison of Approaches to On-Line Handwritten Character Recognition”, *Massachusetts Institute of Technology, PhD Thesis*, 1995.