Improving the Efficiency of a Fuzzy-Based Single-Stroke Character Recognizer with Hierarchical Rule-Base†

A. Tormási* and L.T. Kóczy*
* Széchenyi István University/Department of Automation, Győr, Hungary
tormasi@sze.hu, koczy@sze.hu

Abstract— In this paper we present an improved version of the fuzzy based single-stroke character recognizer introduced in previous works. The modified recognition method is able to reach an acceptable accuracy in the character recognition with a significant decrease on the computational complexity of the algorithm. Different hierarchical rule-base techniques were successfully used to improve the efficiency of fuzzy systems. The altered recognizer reached 98.82% average recognition rate with 26 different single-stroke symbols (based on Palm's Graffiti alphabet) without learning user-specific parameters or modifying the rule-base during the tests. The new algorithm has a small decrease in the recognition rate compared to the accuracy of the original systems but the new method has less computational price than the original system does.

I. INTRODUCTION

There are many problems of usability and ergonomics are posed by the current virtual keyboards. These are using most of the display reducing the maximal size of the content at the expense of the data context which again reduces the user's performance during the workflow.

The best alternative for a keyboard replacement could be handwriting systems. Processing written text by computers nevertheless has a long history. In this field there are still many ongoing research and development projects aiming to achieve more accurate recognition of handwriting. In her study LaLomia determined 97% as the general user acceptance rate for handwriting recognizers [1].

It is important for industrial users and end-users to improve the recognition rate of handwriting recognizers with the smallest possible increase of the computational complexity even if the growth of hardware performance is faster than the growth of resource requirement of the recognition method.

In this paper we present an improved version of a fuzzy based single-stroke character recognizer [FUBAR]. The modified system has reached 98.82% average recognition rate with 26 different single-stroke symbols based on Palm's Graffiti alphabet. The accuracy of the new method is close to the results of other commercial and academic recognizers. Beside the high recognition rate the developed method has much lower computational complexity than the investigated algorithms which makes FUBAR a notable competitor of other character recognition systems.

This paper consists of five sections: After the Introduction in Section 2 the basic concept of our original character recognizer is presented. In Section 3 the steps of rule-base preparation and the modifications made on the original recognition method are proposed. The test results and a comparison with other known recognition systems are introduced in Section 4. In the last section observations are summarized, future works and other possible applications of the system approach are discussed.

II. THE CONCEPT OF THE ORIGINAL RECOGNITION

During the design of the original recognition algorithm [2] besides on acceptable recognition accuracy three main goals were targeted. The first objective was to limit the resources needed for the method as a basic requirement for the use of the system in portable devices. A solution has been worked out to eliminate geometrical transformations from the method so we could reduce the overall computational complexity. A simple and fast exchangeability of the symbol set was our second goal for a better user friendliness. The last objective was to design an easily modifiable symbol-base of writing for the adaptation step which makes our recognizer able to learn the user specific style. In the early stage of the development we decided to handle the stroke segmentation as a separate problem so we could focus on the concept of the recognition engine. The basic concept of FUzzy BAased Recognizer is shown in Fig. 1.

![Figure 1. Model of FUBAR](image.png)
The system uses a modified Palm Graffiti alphabet (strokes for the letter G and U are changed) to make the single-strokes more similar to the Hungarian style of capital letters. The two alternation introduced can be seen in Fig. 2.

![Figure 2. The Altered Symbols in Graffiti and in FUBAR](image)

### A. The Input Signal

Each individual stroke could be represented as a three-dimensional continuous function sampled by the digitizer tablet (Fig. 3). The system collects all the coordinates in chronological order representing the digital ink.

![Figure 3. Input signal from different angles](image)

Due to hardware bottlenecks the input device is not capable of collecting all parts of the signal and the distance between the sampled points may differ (Fig. 4). The range between sampling points depends on the writing speed and the available hardware resources.

![Figure 4. Sampled input signal with varying point density](image)

The varying distance between points renders more difficulty in recognizing the stroke by the average density of points in the extraction grid. To solve this problem re-sampling is unavoidable and it is done in the following (Pre-processing) subsection.

### B. Pre-Processing

This step fixes the density of points (Fig. 5).

![Figure 5. Re-sampled input stroke](image)

It also works as anti-aliasing making the strokes more readable.

The first and last points of the stroke are kept for reference. After that a filtering algorithm calculates the distance between the last added (or first) points and the following ones. If the distance reaches the minimum threshold then the point will be added to the re-sampled stroke.

To calculate the distance between the points the method uses Euclidean distance.

Equation (1) represents the filtering algorithm.

\[
\min_{l} \{ \| l' - \gamma \|_{\dim} \} = \arg \min_{l} \{ \| l' - \gamma \|_{\dim} \} ,
\]

where \( l \) is the list of collected points, \( l' \) is the list of re-sampled points and \( \gamma \) represents the minimum distance between points and \( \arg \min \) gives a minimal position \( a \) at which \( f \) is minimized.

### C. Feature Extraction

Input signals are identified by the width/height ratio of the stroke and the average number of stroke-points in the rows and columns of the fuzzy grid drawn around the input stroke.

The first system used crisp grids (with sharp borders) but tests pointed out that if the angular offset of the input stroke and the etalon symbol were different then distribution of the points in the grid would also differ as shown in Fig. 6. This might cause a considerable reduction in recognition rates.

![Figure 6. Model of FUBAR](image)

As a solution we designed a grid with blurred boundaries which will be referred to as fuzzy grid (each grid row and column defined as a fuzzy set [3] constituting Ruspinian-partitions [4]). In the case where a point is located close to a boundary it will be counted as a member of both columns and rows with different membership degrees calculated by the exact location as shown in Fig. 7.
Point distributions in the fuzzy grid are calculated by the following two formulas:

\[ c_i = \frac{\sum_{j=1}^{\text{dim}^l} \mu_{\text{cell}}(r_j)}{\text{dim}^l}, \quad r_i = \frac{\sum_{j=1}^{\text{dim}^l} \mu_{\text{cell}}(c_j)}{\text{dim}^l}, \]

where \( c_i \) represents the point distribution in the column \( i \) and \( r_i \) represent the same for the rows, \( l' \) is a list of the filtered stroke-points, \( x_i \) and \( y_j \) are the \( x \)- and \( y \)-coordinates of element \( j \).

D. The Inference

Each symbol in the set is represented by a single fuzzy rule [5-6] and an \( N_{\text{sample}} \) sized FIFO queue which stores symbol samples previously written by the user and used only during the adaptation step.

Previously collected stroke-features are used as input parameters for the fuzzy rules. Each rule is evaluated with the features of the current input stroke. The number of rules is equivalent to the number of the symbols in the base set.

Between the parameters of the rule we use min t-norms as AND operators. The consequent part of the rules represents the degree of matching the parameters of the input stroke and the parameters of the symbol represented by the given rules. For the inference we use the Takagi-Sugeno method [6] and the best fitting rule (with the highest rule match) will be chosen as the output of the inference.

E. The Adaptation Phase

After evaluating the inference the next phase is the supervised adaptation. The user has to set the target symbol which he/she wanted to enter via the recognition interface. Using the features of the input stroke and the previously stored samples the system tunes the parameters of the fuzzy sets in the rule base.

During adaptation the system must consider the features of stored samples as much as the new input stroke itself. All parameters of the new symbol must fit to the tuned fuzzy set of the target symbol as much as it is possible without decreasing the fitness of the stored samples. At the same time the method has to minimize the overlap of the target and non-target fuzzy rules without modifying the fitness of the samples stored in the non-target symbols.

To reduce the computational complexity of the system it has been decided to use an evolutionary method [7] for the adaptation process. Evolutionary algorithms are able to provide acceptable (sub)optimal solutions for many problems in a short time with smaller resource requirement than exact optimum search. In the developed algorithm classic evolutionary solutions cannot be used due to the special constraints for the different dynamic fitness functions that have to be applied for the symbols at the same process. Without these constraints the fuzzy sets would overlap in the different rules which would decrease the recognition rate.

As a solution to the overlap problem we extended the bacterial evolutionary algorithm [8] with "punish" and "reward" options. The method rewards the target symbol and punishes all other, non-target symbols by using different fitness functions containing the special constraints. The reward fitness function maximizes the recognition rate for the input symbol and the user-samples stored by the target symbol while minimizes the recognition rate for the non-target symbol samples. The punish fitness function maximizes the recognition rate for the stored user-samples in the current symbol and minimizes the recognition rate for the input symbol and for the stored user-samples from other symbols.

The algorithm consists of the following 4 phases:

Each membership functions of rules are converted into four numbers per function (and nine functions per rule) and stored in an individual vector which represents the chromosomes of a given bacterium (Fig. 8). At the start of this step each symbol is represented by one bacterium which represents the rule of the symbol. Each colony is created by an individual bacterium.

Figure 8. Converting a rule into bacterium

The bacterium in the colonies are replicated \( N_{\text{replicate}} \) times. A colony with the original bacterium and the replicates are representing the initial population. Previously collected symbol-samples are also stored in the colonies for further use.

Bacterial mutation is an operator used on each bacterium in a population separately. A bacterium (selected from the population) is cloned \( N_{\text{clone}} \) times. Next step of this phase is selecting randomly a chromosome and replacing it in the clones except in the original bacterium with a random value (mutation) as shown in Fig. 9.

Figure 9. Concept of bacterial mutation

After modifying a chromosome we have to check the validity of the bacterium by inspecting the chromosomes.
Points of a trapezoidal membership function represented by four chromosomes must be ordered by value as presented in Fig. 10.

Figure 10. Determining validity of series of chromosomes

If the chromosomes are not in a good order then the algorithm fixes it. The algorithm orders the original bacterium and all its clones by fitness value and chooses the best one which replaces the selected chromosome in the others (Fig. 11).

Figure 11. Bacterium chromosome transfer

This procedure (cloning – mutation – test – evaluation – replacement) is repeated until all the chromosomes have not been chosen as seen on Fig. 12. After selecting all the chromosomes the algorithm puts the best bacterium back into the population and the others will be deleted. As a result the new bacterium will have a better or at least the same fitness value (in case of unavailing mutation).

These steps are repeated in populations until all the bacteria have not been chosen.

Figure 12. Schematics of bacterial mutation

During this phase the gene transfer operator is used on all the populations (with mutated bacterium) separately. The method chooses a population ordered by fitness value and divides it into two pieces, the group of the good and the group of bad bacteria.

One bacterium is chosen from the good ones (let us call it "source bacterium") and another one from the bad ones (let us call it "target bacterium"). The source bacterium replaces a randomly chosen chromosome in the target bacterium (gene transfer) in the third step as shown in Fig. 13.

Figure 13. A good bacterium infects a bad bacterium

As described in Phase 3, after the modification of a chromosome the membership functions represented by the bacterium must be validated (and adjusted if it is necessary). The algorithm repeats this phase \(N_{inf}\) times which stands for the maximal number of infections. After using gene transfer on all the populations we have to increase \(N_{gen}\) which indicates the number of the current generation. If \(N_{gen} < N_{max-gen}\) (maximal number of generations) then repeat the algorithm from Phase 3 else choose the best bacterium from all the colonies, convert them back to symbols and replace the original symbols with the new ones. The complete phase is presented in Fig. 14.

Figure 14. Steps of gene transfer

III. THE DEVELOPMENT OF HIERARCHICAL RULE-BASES

There are numerous works on the application of hierarchical rule-bases in fuzzy systems [9, 10, 11] in which the results showed that the computational complexity might be decreased of different fuzzy systems.

The original system’s rule-base has been used to build up the hierarchy shown on Fig. 15. Each structure is based on only one rule-input dimension. The overlapping membership functions are representing the different groups of sub rule-bases and each sub rule-base has a meta-rule. The different groups are evaluated only if the associated meta-rule reached the highest result.

Figure 15. Schematics of bacterial mutation
The recognition method with a 6-by-6 fuzzy grid recognizes 9,942 from 10,000 input symbols (99.42%) [12]. The same system with a 6-by-4 fuzzy grid recognizes 11,727 letters from 11,818 (99.23%) for the same computational price [13]. Considering the previous researches, the computational complexity and the accuracy of the method, a recognizer with a 6-by-4 fuzzy grid have been chosen as the basic system for the improvement.

All the possible hierarchies have been built up based on the different rule input parameters but only the (almost) balanced hierarchies have been tested. The system with the hierarchy based on “Row #3" input parameter reached the best average recognition rate shown in Fig. 16. The 98.82% average accuracy was reached which is lower than the accuracy of the original system but the computational cost is much lower in the new method.

In some cases the membership functions were overlapping a way too much which would cause at least as complex rule-base as the original one. The groups were containing too much or only a few symbols which would lead us to lower accuracy and efficiency. In other cases the decrease in recognition rate was higher than the decrease in the computational complexity as seen in Fig. 17.

With different hierarchies (based on multiple parameters) or other methods such as antecedent weighting and adaptation the recognition rate might be increased. The algorithm reached 98.82% recognition rate which is over the user acceptance threshold and it has lower computational complexity than any other known method which makes the new algorithm usable (e.g. in automated form or postal letter processing methods).

### IV. RESULTS

The original system reached 99.23% average accuracy rate using a 6-by-4 fuzzy grid with 26 different symbols. The recognition rates for individual symbols are shown in Table 1.

![Table 1: Average Recognition Rates of FUBAR with a 6-by-4 Fuzzy Grid](image)

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Average recognition rate (%)</th>
<th>Symbols</th>
<th>Average recognition rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>100</td>
<td>N</td>
<td>100</td>
</tr>
<tr>
<td>B</td>
<td>92.7377</td>
<td>O</td>
<td>93.8889</td>
</tr>
<tr>
<td>C</td>
<td>97.7777</td>
<td>P</td>
<td>100</td>
</tr>
<tr>
<td>D</td>
<td>98.8889</td>
<td>Q</td>
<td>100</td>
</tr>
<tr>
<td>E</td>
<td>98.8889</td>
<td>R</td>
<td>100</td>
</tr>
<tr>
<td>F</td>
<td>100</td>
<td>S</td>
<td>100</td>
</tr>
<tr>
<td>G</td>
<td>100</td>
<td>T</td>
<td>100</td>
</tr>
<tr>
<td>H</td>
<td>100</td>
<td>U</td>
<td>100</td>
</tr>
<tr>
<td>I</td>
<td>100</td>
<td>V</td>
<td>100</td>
</tr>
<tr>
<td>J</td>
<td>100</td>
<td>W</td>
<td>100</td>
</tr>
<tr>
<td>K</td>
<td>99.4444</td>
<td>X</td>
<td>98.3333</td>
</tr>
<tr>
<td>L</td>
<td>100</td>
<td>Y</td>
<td>100</td>
</tr>
<tr>
<td>M</td>
<td>100</td>
<td>Z</td>
<td>100</td>
</tr>
</tbody>
</table>

With hierarchical rule-base the same system has a 98.82% average recognition rate. The recognition rates for letters are shown in Table 2.

![Table 2: Average Recognition Rates of FUBAR with a 6-by-4 Fuzzy Grid and Hierarchical Rule-Base](image)

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Average recognition rate (%)</th>
<th>Symbols</th>
<th>Average recognition rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>100</td>
<td>N</td>
<td>96.0894</td>
</tr>
<tr>
<td>B</td>
<td>92.7374</td>
<td>O</td>
<td>93.8548</td>
</tr>
<tr>
<td>C</td>
<td>97.7654</td>
<td>P</td>
<td>100</td>
</tr>
<tr>
<td>D</td>
<td>98.8827</td>
<td>Q</td>
<td>100</td>
</tr>
<tr>
<td>E</td>
<td>97.2067</td>
<td>R</td>
<td>100</td>
</tr>
<tr>
<td>F</td>
<td>100</td>
<td>S</td>
<td>100</td>
</tr>
<tr>
<td>G</td>
<td>100</td>
<td>T</td>
<td>100</td>
</tr>
<tr>
<td>H</td>
<td>100</td>
<td>U</td>
<td>97.2067</td>
</tr>
<tr>
<td>I</td>
<td>100</td>
<td>V</td>
<td>98.324</td>
</tr>
<tr>
<td>J</td>
<td>100</td>
<td>W</td>
<td>100</td>
</tr>
<tr>
<td>K</td>
<td>99.4413</td>
<td>X</td>
<td>98.324</td>
</tr>
<tr>
<td>L</td>
<td>100</td>
<td>Y</td>
<td>99.4413</td>
</tr>
<tr>
<td>M</td>
<td>100</td>
<td>Z</td>
<td>100</td>
</tr>
</tbody>
</table>
The average accuracy of the modified system is lower than the original system's one but the computational complexity of the algorithm is outstanding compared to other recognizers. The reached recognition rates are presented in Fig. 18.

![Figure 18. Average recognition rates of different recognition methods](image)

In the study of M. D. Fleetwood et al. [14], they compared the performance of Palm virtual keyboard and Graffiti recognizer. The users reached 98% accuracy with the keyboard and 91% recognition rate with the Graffiti single-stroke recognizer.

In another study T. Költringer and T. Grechenig [15] analyzed the performance of the improved Graffiti (also known as Graffiti 2) which was able to recognize few multi-stroke gestures too (representing i, k, t and x characters). Test results showed 86.03% recognition rate.

The $1 recognizer presented by J. O. Wobbrock et al. reached 97% accuracy using 16 gestures with one loaded template per symbol [16]. Our system reached 99.97% recognition rate with 16 symbols using only one rule per symbol.

SN (the improved $1) recognizer reached 93.7% accuracy using 20 multi-stroke symbols with more than 3 loaded templates per gesture [17].

V. CONCLUSION AND FUTURE WORK

The developed system reached a recognition rate that is well over 97% which is the user acceptance threshold defined by LaLomia.

The used algorithm does not contain difficult geometrical transformations such as scaling and rotating to save resource and time. The computational complexity of the pre-processing in FUBAR is $O(N_p)$ where $N_p$ is the number of points representing the input stroke (with the order of 100). During the feature extraction phase the algorithm complexity is $O(N_f)$ where $N_f$ stands for the number of points in the stroke after resample phase (with the order of 10). In the inference phase the complexity is $O(N_d + N_m + N_o)$, where $N_d$ is the number of meta-rules, $N_m$ is the number of symbols in a given branch of hierarchy and $N_o$ is the dimension of fuzzy rules (in our tests it is 11 for the symbol rules). This is less than the computational complexity of other investigated recognition systems.

The use of hierarchical rule-base in FUBAR resulted in a high decrease in the computational complexity of the algorithm. The new method has reached the same recognition rate as all other published academical and commercial recognition systems and the computational complexity is much lower than in all other analyzed systems.

With different improvements the developed system could have a lower computational complexity and a greater recognition rate.

Currently we are analyzing the effect of different hierarchical rule-bases on recognition rates and computational complexity.

Different methods are under development to support multi-stroke character recognition.

The capability of off-line character recognition is needed for further tests on public character bases and for possible industrial applications. This could be done by adding a "Phase 0" to the algorithm which provides usable stroke information from image input.

It is also important to analyze the accuracy of the developed method using different alphabets (e.g. Greek).

We have plans to add a dictionary-like support to the recognition algorithm. Currently it does not use any word or letter based support which could increase the recognition rate.

REFERENCES


