Available online at www.sciencedirect.com
ScienceDirect
Engineering Applications of Artificial Intelligence 21 (2008) 952–960
Engineering Applications of
ARTIFICIAL
INTELLIGENCE
www.elsevier.com/locate/engappai

A video-based text and equation editor for LaTeX

Özcan Öksüz[a], Uğur Güdükbay[a,*], A. Enis Çetin[b]

[a]Department of Computer Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey
[b]Department of Electrical and Electronics Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey

Received 29 August 2006; received in revised form 18 July 2007; accepted 7 August 2007
Available online 5 November 2007

Abstract

In this paper we present a video based text and equation editor for LaTeX. The system recognizes what is written onto paper and generates the LaTeX code. Text and equations are written on a regular paper using a board marker, and a USB camera attached to a computer is used to capture and record the pen-tip positions in each consecutive image frame. Characters and symbols are represented as separate finite state machines (FSMs). They are written in an isolated manner and they are recognized on-line using the FSMs. In the last step, LaTeX code corresponding to recognized characters and symbols is generated.

Keywords: Pattern recognition; Handwritten character recognition; On-line recognition; Mathematical notation recognition; Mathematical expression recognition; LaTeX

1. Introduction

In this paper, we present a video based text and equation editor for LaTeX Lamport (1999). In this system, each character or symbol is represented as a separate finite state machine (FSM). The user writes text and equations on a paper and a camera attached to a computer records the actions of the user. Written characters are recognized on-line and the corresponding LaTeX code is generated at the end of each page.

In our system, the user writes on regular paper or a white board and a USB camera attached to the computer captures the image frames while he or she writes text and equations. We detect the pen-tip positions using the captured image frames and store them in a linked list. After the user finishes writing, he or she starts the recognition and LaTeX code creation procedure. This procedure is explained in the sequel.

First, we convert the picture frame at the end of the video sequence into a binary image and find the bounding boxes of each character or symbol. Then, we calculate the chain codes, from information about changes in direction of the pen tip, and the region codes, from the position of the pen tip within the bounding box sub-rectangles. Then characters and symbols are recognized by feeding the calculated chain and region codes to the FSM in the training set. Finally, LaTeX codes corresponding to the recognized characters and symbols are generated.

The contributions of the paper can be summarized as follows:

1. The main contribution of the paper is a framework for automatic LaTeX code generation. The proposed framework uses an on-line character recognition approach based on a pen-input interface. The interface uses a standard USB camera and pen and paper to input user-specified text. The system recognizes the text on-line based on pen movements and converts it into LaTeX code automatically. To the best of our knowledge, this is a first attempt to generate LaTeX code automatically based on hand-written user input. In the current implementation, the framework requires some special marks in the input to specify some of the LaTeX

*Corresponding author. Tel.: +90 312 290 13 86; fax: +90 312 266 40 47.
E-mail addresses: oksuz@cs.bilkent.edu.tr (Ö. Öksüz), gudukbay@cs.bilkent.edu.tr (U. Güdükbay), cetin@ee.bilkent.edu.tr (A.E. Çetin).

0952-1976/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2007.08.003

commands. However, we plan to improve the implementation so that the LaTeX code is generated directly from the user input that does not contain any special marks to specify LaTeX commands.

2. The proposed character recognition algorithm uses both region and chain codes. Unlike the previous methods that use Graffiti™-like characters, we use the actual representations of the characters in the English alphabet. In other words, our character set is not a subset of the alphabet and the user do not need to memorize artificially created characters to write documents. In the existing methods, characters are specially chosen so that their chain codes are guaranteed to be different. That is, the chain codes are enough to separate and recognize each character in their character set. However, we are using regular alphabet characters and some of the character share the same chain codes. For example, chain codes of character c is included in chain codes of character e. In order to differentiate and improve recognition rate, we propose a new method that incorporates region codes in addition to chain codes.

The rest of the paper is organized as follows: related work in the area of handwritten character recognition and the recognition of handwritten mathematical notation is presented in Section 2. In Section 3, key features of our character recognition system are explained. LaTeX code generation process is explained in Section 4. Conducted experiments and their results are presented in Section 5. Finally, Section 6 contains conclusions and suggestions for future work.

2. Related work on handwriting recognition

Handwriting recognition can be classified into two categories: *on-line* and *off-line* (Plamondon and Srihari, 2000). These differ in the form the data is presented to the system. In off-line systems, something is written on a paper; this paper is then digitized and the writing is decoded (Arica and Yarman-Vural, 2001). In on-line systems, however, the handwriting is recognized while it is being produced.

On-line recognition has some advantages over off-line recognition, mostly related to the increased amount of information that can be obtained. This is information about the spatial properties of the stylus, the number of strokes, their order, the direction of writing for each stroke, and the speed of writing. Another advantage of on-line recognition is that there is a close interaction between the user and the machine. Thus, the user can correct any recognition error immediately when it occurs. Moreover, on-line handwriting recognition promises to provide a dynamic means of communication with computers through a pen-like stylus, rather than just a keyboard. This seems to be a more natural way of entering data into computers (Bunke et al., 1999; Wienecke et al., 2001).

In the literature, many different handwritten character recognition methods are proposed and implemented. Tang and Lin (2002) developed a robust stroke-tracing algorithm for a video-based recognition system for handwritten Chinese characters. Their algorithm works effectively against various shadow and noise problems. They accurately extracted the temporal stroke information in a fashion similar to an on-line OCR system. Wienecke et al. (2001) proposed a complete recognition system based on visual input. Acquired images are processed for pen tracking, then features are extracted and recognition is performed using hidden-Markov-models.

The trajectory of the pen tip is the essential information for on-line handwriting recognition. Therefore, the position of the pen tip has to be determined in every frame of the video sequence. In general, special devices are necessary to obtain the pen trajectories. For example, a laser pointer is a robust text entry device in changing lighting and background conditions and it was used by Özer et al. (2001). A video-based data acquisition approach was proposed by Munich and Perona (1996); in their method, ink traces left on the paper while written characters are extracted automatically using difference images.

The use of FSM in character recognition is investigated by many researchers. Özer et al. (2001) proposed a vision-based system for recognizing isolated Graffiti™ characters. In their system, recognition is performed by a bank of FSMs whose input is the chain code of the hand drawn character. Erdem et al. (2004) extended the work, moving from isolated Graffiti™ recognition to continuous Graffiti™ recognition. They used a modified version of the Graffiti™ alphabet and obtained a word recognition rate of 93%.

2.1. Recognition of handwritten mathematical equations

There are many different approaches to recognition of handwritten mathematical (Blostein and Grbavec, 1997; Chan and Yeung, 2000). Some important problems that should be addressed in recognition of handwritten mathematical formulas are (Zanibbi et al., 2002) locating the mathematical expressions in a document image (Kacem et al., 2001; Fateman, 1999), recognition of large number of mathematical symbols that uses many different fonts, typefaces, and sizes, distinguishing between noise and small symbols such as periods and commas (Berman and Fateman, 1994), distinguishing between inline, superscript and subscript relations (Wang and Faure, 1988) (this is also related with bad writing habits such as wrong symbol placement), resolving ambiguous spatial relationships and symbol identities using contextual analysis (Miller and Viola, 1998; Zanibbi, 2000) (this is even more difficult in online recognition systems), and different handwriting styles for mathematical notation.

The mathematics recognition system described by Zanibbi et al. (2002) is an offline recognition system that makes three successive passes over the input and uses tree transformation approach. They match the mathematical

expressions to tree structures and convert these trees to LaTeX expressions. Our system, on the other hand, is an online recognition system that uses pen-tip trajectory for recognition.

Smithies et al. (1999) presented a system for editing equations based on handwritten input. They use an on-line recognition algorithm based on nearest-neighbor classification. The systems described in Toyozumi et al. (2001) and Garain and Chaudhuri (2004) are also examples of online recognition systems for mathematical expressions. Toyozumi et al. use Freeman chain code to recognize strokes and combine several strokes into a character based on their positions and combinations. Then, they make a structural analysis by dividing the formula into blocks. Garain and Chaudhuri uses a two stage approach, namely symbol recognition and structural analysis. They use online and offline features together to identify the spatial relationships among symbols. Fukuda et al. (1999) and Sakamoto et al. (1998) use directional approaches to capture writing directions of strokes for character recognition. Other notable examples of studies on recognition of mathematical notation are Anderson (1977), Chan and Yeung (1999, 2000, 2001), Chou (1989); Chang (1970); Okamoto and Miao (1991); Phillips and Chhabra (1999); Eto and Suzuki (2001); Kanahori and Suzuki (2001); Winkler et al. (1995); Faure and Wang (1990); Grbavec and Blostein (1995).

3. Character recognition system

Our system lets users write on paper as in an off-line system. However, instead of waiting for a whole page to be written and then scanned into a computer, a USB Camera attached to the computer is used to capture a sequence of the character images while they are being written. Next, the binary image is obtained using thresholding and the bounding boxes corresponding to each written character or symbol are extracted. Then, chain and region codes are calculated and fed into the FSMs in the training set. The character or symbol whose FSM generated the smallest error is given as the recognized result. Finally, LaTeX codes are generated using the recognition results.

Key features of our character recognition system are described in following subsections.

3.1. Pen trajectory and chain codes

The trajectory of the pen tip is the essential information for on-line handwriting recognition. The user writes the characters using an ordinary black board marker. A blue band is attached near the pen tip and the position of this blue band is recorded at each frame of the video sequence. After the user has finished writing and has started the recognition process, the pen trajectory points are processed for chain code extraction. A chain code is a sequence of numbers between 0 and 7 obtained from the quantized angle of the pen tip's point recorded at fixed intervals.

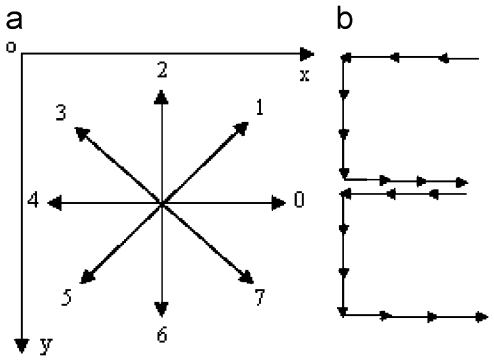

Fig. 1. Chain codes. (a) chain codes for different directions, (b) chain coded representation of character E = 444666000444666000.

Chain code values for angles and chain coded representation of the character 'E' are shown in Fig. 1.

In our system, each symbol is represented with several different chain code sequences. For instance, chain code sequences for the character 'N' is shown in Fig. 2. By this way, our system can handle variations in the writing styles and different users can adopt to our system easily. Therefore, users need little training to use the system.

Chain codes show the writing direction for the character and they are used for character representation and recognition. One requirement in our system is that each symbol must be represented with at least two different chain code values; in other words, each symbol must have at least two strokes. Since '.' and ',' cannot be represented with at least two chain code values, we used different symbols in place of these punctuation marks. We used '△' to represent '.' and we used 'Λ' to represent ','.

3.2. Bounding boxes and their sub-rectangles

When the user starts the character recognition and the LaTeX code generation routine, all the pen trajectory points for all the letters written are in the picture frame at the end of the video sequence. We need to separate the trajectory points of each letter from the rest of the points.

First of all, the coordinates of the pen trace points are corrected because there is a gap between the blue band attached to the pen and the pen tip. Then, the stored picture frame is converted into a binary image using thresholding. Next, this binary image is processed to determine the bounding box of each character or symbol; this is the minimum rectangle enclosing the written character or symbol. For each bounding box, we start from the first point of the linked list holding the pen-tip trajectory points and test whether this point lies inside that bounding box or not. If it is not in the bounding box, we continue to the next point and test again until we find a point that lies inside it. This first point lying inside of the bounding box is marked as the 'entrance' point. After the entrance point is marked, we continue testing the rest of the points in the sequence. The first point lying outside of the bounding box is marked as the 'exit' point. Thus the

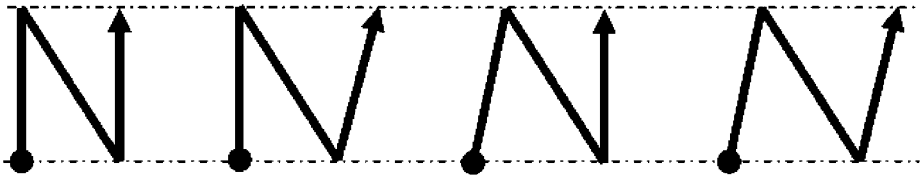

Fig. 2. Chain code sequences for character 'N'.

points between the entrance and the exit points are the trajectory points belonging to the character or symbol inside the current bounding box. In this way, the trajectory of each written character or symbol is separated from all the other trajectory points and corresponding chain codes are generated.

Each symbol is represented by several different chain code sequences and some symbols cannot be differentiated just by considering chain codes. For example, if the user writes 'N', the system returns both 'N' and 'Λ' as the recognition outcome only when the chain code values are used. For that reason, separated pen movement points are further investigated and their relative positions in the bounding box are used during the classification period. In order to use the positional information, each calculated bounding box is divided into sub-rectangles. Bounding boxes for character group 1 (A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, b, d, f, g, h, j, k, l, p, q, y, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, (,), [,], {, }, √, ∑, ∏, ∫, Δ, λ, Θ, γ, δ, β, Ω, Φ) are divided into 9 different sub-rectangles, and bounding boxes for character group 2 (a, c, e, i, m, n, o, r, s, u, v, w, x, z, *, +, /, <, >, =, '.', ',', \, π, θ, α, ε, σ, ^) are divided into five sub-rectangles, as shown in Fig. 3. Then a region-coded representation of each character is determined such that each separate character stroke is described with the minimum number of bounding sub-rectangles, as shown in Fig. 4. If one stroke cannot be bounded within one sub-rectangle, sub-rectangles are merged until the stroke is bounded. Therefore each character is assessed using both chain and region codes. Consequently, if the user writes 'N', the system uses both chain and region code values to identify it only as 'N'.

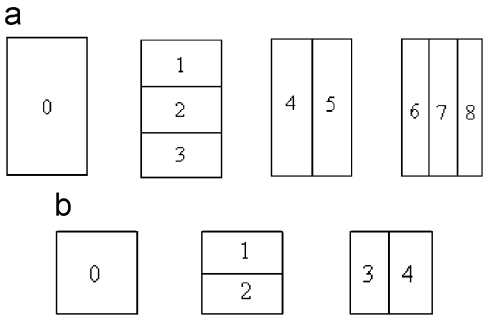

Fig. 3. Bounding box sub-rectangles, and their preclassification groups. (a) group 1 (b) group 2.

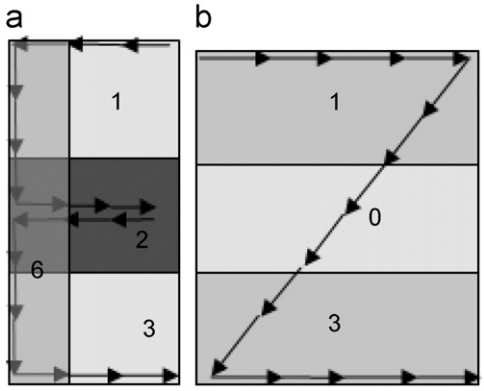

Fig. 4. Region coded representation. (a) $E = 162263$, (b) $Z = 103$.

3.3. Finite state machines

The recognition system uses FSMs corresponding to individual characters and symbols. FSMs for new characters or symbols are easily defined and added to the system. As an example, we will create the FSM of the character 'N'. From different writing styles of 'N' (see Fig. 2), we see that the first stroke of the letter follows direction 1 or 2 of the chain code and it is completely bounded by the subrectangle 4, as shown in Fig. 5(a). Therefore the state of the first stroke is "$C = 1, 2 + R = 4$" where 'C' represents the chain code and 'R' represents the region code. For the second stroke, we see that it follows direction 7 of the chain code and its minimum bounding subrectangle is the whole bounding box which is represented by subrectangle 0, as shown in Fig. 5(b). So, the state for the second stroke is "$C = 7 + R = 0$". The chain code of the last stroke is the same as the first one, namely 1 or 2. However, the region code is 5 for that stroke, as shown in Fig. 5(c). Therefore, the state of the last stroke is "$C = 1, 2 + R = 5$". The FSM for the character 'N' is shown in Fig. 6(a). Similarly Fig. 6(b) shows the FSM corresponding to '∑'.

Separated chain codes and calculated region codes are inputs to the FSMs in the training set. The weighted sum of the error from each state determines the final error for a character in the recognition process. The FSM generating the minimum error identifies the recognized character or symbol.

An example of how FSMs work in our system follows. When the 23222217771112 chain code and the 44444440005555 region code are applied as an input to N's machine, the first elements, chain 2 and region 4, are correct values at the FSM's starting state. Therefore, the

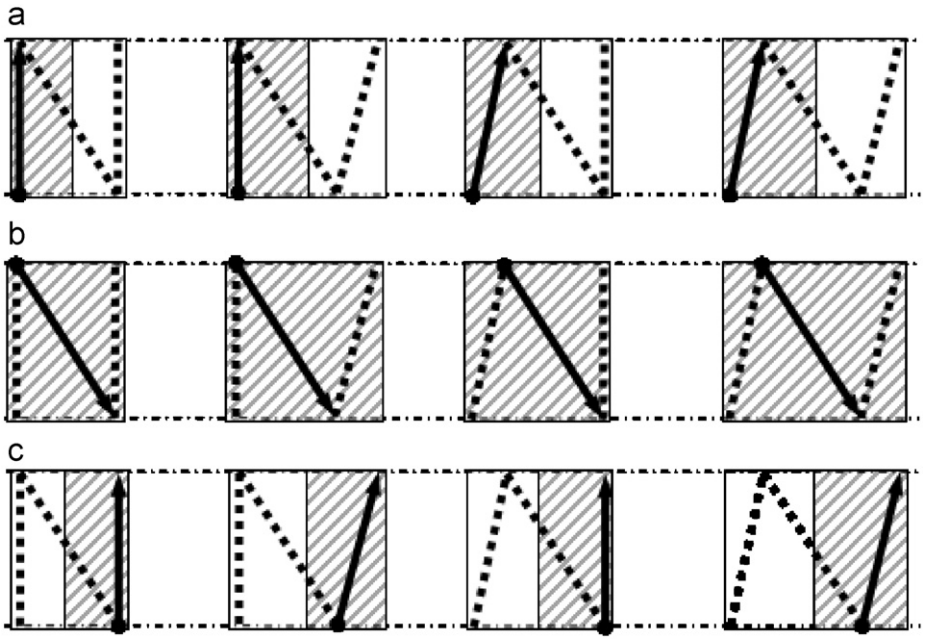

Fig. 5. Individual states of 'N', where C = Chain Code, R = Region Code. (a) C = 1, 2 + R = 4, (b) C = 7 + R = 0, (c) C = 1, 2 + R = 5.

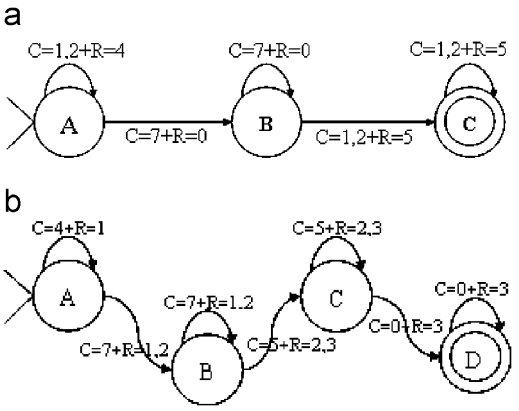

Fig. 6. Finite state machines of (a) N and (b) $\sum$, where C = Chain Code, R = Region Code.

FSM is started with error count 0. The second element, chain 3, generates an error and the error counter is set to 1. The FSM remains in the first state for the other 2s of the chain and the 4s of the region code values. It remains in the same state with the subsequent 1 since 1 and 2 are the inputs of the machine's first state for N. Input of chain code 7 and region code 0 makes the FSM go to the next state, and the subsequent chain codes of three 7s and region codes of three 0s make the machine remain in that state. Whenever the chain code becomes 1 and the region code becomes 5, the FSM moves to the final state. The rest of the input data, chain codes 1 and 2 with region code 5, makes the machine stay in the final state; and when the input is finished, the FSM terminates. For this input sequence, the machine's error for character N is 1. However, the other FSMs generate either greater or infinite error values for this input.

```
- locate the reference lines
- calculate the bounding boxes
for each bounding box
  - separate pen-tip points inside it
  - construct chain codes
  - calculate sub-rectangles
  for each character in the total character set
    while character states and pen-tip
          points are not finished do
      if chain code equals to
         character state and
         point is in state rectangle
        - move to next state
      else
        - increase recognition error
        - move to next point
      end
    end
    if character states not finished
      - set error to infinity
  end
  - post process the results
  - find the best match
end
- generate LaTeX Code
```

Fig. 7. The character recognition algorithm.

3.4. Steps of the character recognition algorithm

While characters are being written in each frame, the position of the blue band attached to the pen tip is stored in

a linked list. After characters are written, the last image of the video is converted into a binary image using thresholding. Then the character recognition algorithm is applied to the binary image. The algorithm is given as pseudo-code in Fig. 7. The overall recognition and LaTeX code creation takes an average of 0.3 seconds in our test PC.

4. LaTeX code generation

The last step in our system is the generation of LaTeX codes corresponding to the recognized characters. Our LaTeX code generation routine is based on assigning character combinations to LaTeX keywords. A limited number of character combinations are used and some of the character combinations have different effects in different environments. The following LaTeX environments are supported: *array construction*, *citation*, *section*, *itemization*, *equation* and *normal text environment* Lamport (1999). Character combinations and corresponding LaTeX codes are shown in Table 1.

To create an array, the user first writes "<a" to indicate the start of the array. Then he writes the column value; next, he writes "," to separate each column of array. In order to move to the next row of the array, he writes "\". Finally, to exit from array creation mode, the user writes ">a". Other supported modes are created in a similar manner and keywords for other modes are shown in Table 1.

When all the recognized characters are in hand, words and character combinations, which are separated from each other by a space character, are stored in an array. Then each array entry is processed sequentially. First, the array entry is compared with the first column values of Table 1. If the entry matches one of the column values, the corresponding LaTeX code is written in the place of the entry and the next entry is processed. If there is no match, the entry is compared with $\int$, $\sum$, $\prod$; if they match, we search the array to find their lower and upper bounds and their arguments. If the entry does not also match with $\int$, $\sum$, or $\prod$, the entry is written as it is.

Table 1
Character groups and corresponding LaTeX codes

Characters	LaTeX code
<a	begin{array}
, in array mode	&
\ in array mode	\\
>a	end{array}
<c	\cite{
>c	}
<e	begin{equation}
>e	end{equation}
<i	begin{itemize} \item
\ in itemize mode	\item
>i	end{itemize}
<s	\section{
>s	}

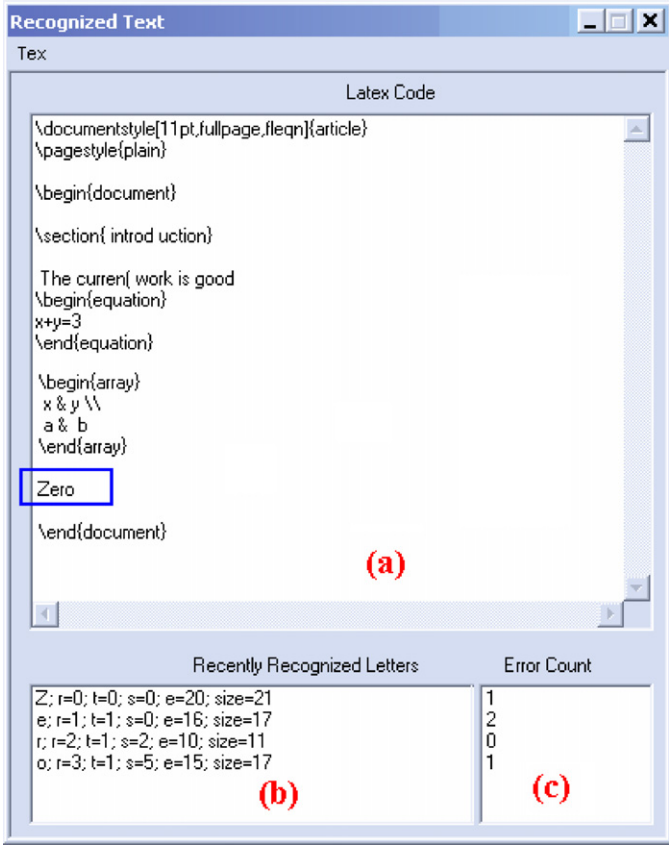

Fig. 8. Recognized text window. (a) generated LaTeX codes, (b) detailed information about the current recognition process, (c) recognition errors associated with each character.

A dialog box is used to present the recognition results to the user and to provide some means of changing the text when required (Fig. 8). It consists of three parts. The editbox holding the generated LaTeX codes is shown in Fig. 8(a). In Fig. 8, the currently recognized word is bounded by a blue rectangle. The user can edit the words in this area when any recognition error occurs. Fig. 8(b) gives detailed information about the current recognition process. The following properties are recorded at each row: recognized character, bounding rectangle index, index of chain in which recognition process started (entrance index), index of chain in which recognition is completed (exit index) and total length of the chain. Fig. 8(c) shows the recognition errors associated with each character. In Fig. 8 the character "Z" is recognized with error = 1 and the character "e" is recognized with error = 2.

5. Experiments

We use an ordinary board marker that is black and we attach a blue band near its tip to detect pen tip movement while characters are drawn. We write characters on white A4 paper or a white board.

We have performed two experiments using two different USB CCD cameras on a test PC with an AMD Athlon 1400 processor having 768 Mbytes of memory. The tests and their results are given below. In both tests we used at least 20 samples for each character and a total of 2170 characters.

- Test 1: We used a USB CCD camera that can capture 30 frames per second with 320×240 pixels. The system handle only 40 characters per minute due to the limitations of the USB camera we use. We attain a 92% recognition rate at a writing speed of about 40 characters per minute.
- Test 2: We used a USB CCD camera that can capture 30 frames per second with 640×480 pixels. Due to high volume of frames available and higher resolution, we obtained a 93% recognition rate at a writing speed of about 40 characters per minute.

The main recognition errors were due to inaccurate writing habits and ambiguity related to similar shaped characters. Most of the confusion was between character pairs such as 'e' and 'c', '5' and 'S', and 'u' and 'v'. This could be avoided by using a dictionary to check for possible character combinations. Contextual knowledge will help to eliminate the ambiguity. For example, if the user wrote "can", our system could recognize the first letter either as 'e' or 'c' with different error counts. If the error count for 'e' is less than error count for 'c', the system would recognize the written word as "ean". However, if a dictionary were included, the system could search all the found character combinations and choose the one in the dictionary with the smallest error count.

Another type of recognition error arises from using a blue band to detect the pen-tip position in each image frame. This blue band may not be visible to the camera at all times due to the writing habits of the user; thus for some frames, pen-tip positions may not be detectable. As a result, the generated chain and region codes may not represent the written character or symbol and it may be either recognized as a different character or not recognized at all. This kind of recognition errors are visible with a USB CCD camera having low capture rates and they are mostly eliminated using a better frame grabber with high

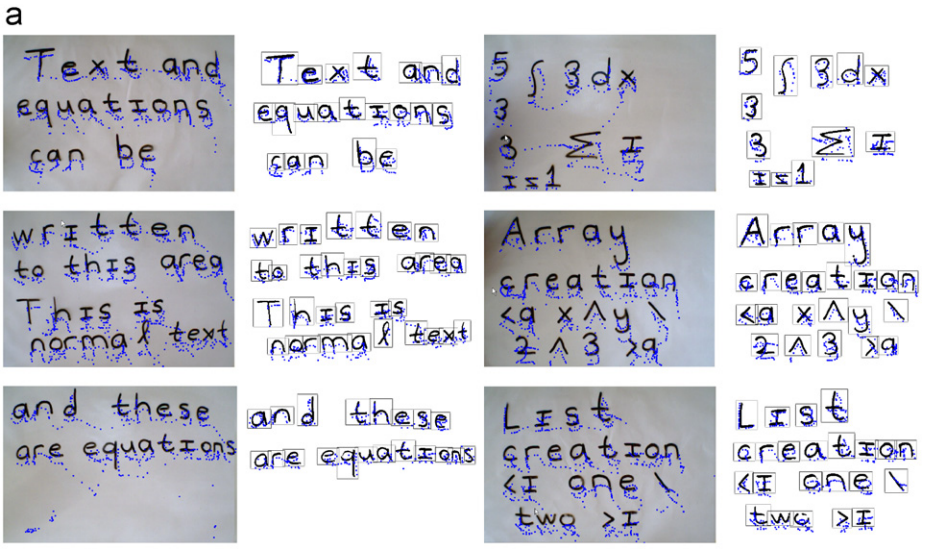

Fig. 9. The character recognition system at work. The images are ordered in a columnwise fashion. (a) written characters, (b) corresponding LaTeX codes.

capture rates. In addition, this type of errors can be further eliminated by using high resolution fram graber and/or detecting pen-tip using difference images Munich and Perona (1996).

In Fig. 9, some examples of written characters and their LaTeX codes are given. Each picture in this figure consists of two columns. The left one is the last frame captured while characters are being written. The pen movement points are shown in blue. The right one shows the binary image with the bounding boxes and pen trace points falling within each bounding box.

6. Conclusion and future work

In this paper, we present a video based text and equation editor for LaTeX. In our system, the user writes text and equations on paper and a camera attached to a computer records the actions of the user. Later, we detect the pen-tips in each image frame and determine the bounding boxes of each character. Next, we find the preliminary classification group of the written characters. Then, we calculate the chain and the region codes. Finally, characters are recognized using FSMs and LaTeX codes are generated.

The experimental results show a 92% correct recognition rate using a USB CCD camera that has a 320×240 capture resolution and a 93% correct recognition rate using a USB CCD camera having a 640×480 capture resolution. Mainly there are two kinds of recognition errors. First kind of recognition errors are due to inaccurate writing habits and ambiguity arising from similar shaped characters. Recognition accuracy can be improved by using a dictionary to check possible character combinations or by using a pop-up menu for confusing characters. The presence of contextual knowledge could help to eliminate the ambiguity. The second kind of recognition errors are caused by using a blue band to detect pen-tip position in each frame. This kind of recognition errors are mostly eliminated using a high-rate frame grabber and they can be further eliminated by detecting pen-tip using difference images.

Forcing the user to enter text and equations in a specific format are the restrictions that limit user flexibility. In future work, we will remove these restrictions step by step and give much more space and freedom to the user. Besides, in order to eliminate some of the recognition errors and have a better recognition rate, we will detect pen-tip positions using difference images.

Acknowledgments

This work is partially supported by European Commission 6th Framework Program with grant number FP6-507752 (MUSCLE Network of Excellence Project). We are grateful to Kirsten Ward for proofreading and suggestions.

References

Anderson, R.H., 1977. Two-dimensional mathematical notations. In: Fu, K.S. (Ed.), Syntactic Pattern Recognition Applications. Springer, New York, pp. 147–177.

Arica, N., Yarman-Vural, F.T., 2001. An overview of character recognition focused on off-line handwriting. IEEE Transactions on System Man and Cybernetics Part C 31 (2), 216–233.

Berman, B.P., Fateman, R.J., 1994. Optical character recognition for typeset mathematics. In: Proceedings of the International Symposium on Symbolic and Algebraic Computation, pp. 348–353.

Blostein, D., Grbavec, A., 1997. Recognition of mathematical notation. In: Handbook of Character Recognition and Document Image Analysis. World Scientific, Singapore, pp. 557–582.

Bunke, H., von Siebenthal, T., Yamasaki, T., Schenkel, M., 1999. Online handwriting data acquisition using a video camera. In: Proceedings of International Conference on Document Analysis and Recognition, pp. 573–576.

Chan, K., Yeung, D., 2000. Mathematical expression recognition: a survey. International Journal Document Analysis and Recognition 3 (1), 3–15.

Chan, K.-F., Yeung, D.-Y., 1999. Recognizing on-line handwritten alphanumeric characters through flexible structural matching. Pattern Recognition 32, 1099–1114.

Chan, K.-F., Yeung, D.-Y., 2000. An efficient syntactic approach to structural analysis of on-line handwritten mathematical expressions. Pattern Recognition 33, 375–384.

Chan, K.-F., Yeung, D.-Y., 2001. Error detection error correction and performance evaluation in on-line mathematical expression recognition. Pattern Recognition 34, 1671–1684.

Chang, S., 1970. A method for the structural analysis of two-dimensional mathematical expressions. Information Sciences 2, 253–272.

Chou, P.A., 1989. Recognition of equations using a two-dimensional stochastic context-free grammar. Visual Communication and Image Processing IV, 852–863.

Erdem, İ.A., Erdem, M.E., Atalay, V., Çetin, A.E., 2004. Vision-based continuous graffiti-like text entry system. Optical Engineering 43 (3), 553–558.

Eto, Y., Suzuki, M., 2001. Mathematical formula recognition using virtual link network. In: Proceedings of the Sixth International Conference on Document Analysis and Recognition, pp. 762–767.

Fateman, R.J., 1999. How to find mathematics on a scanned page. Proceedings of SPIE 3967, 98–109.

Faure, C., Wang, Z.X., 1990. Automatic perception of the structure of handwritten mathematical expressions. In: Plamondon, R., Leedham, C.G. (Eds.), Computer Processing of Handwriting. World Scientific, Singapore, pp. 337–361.

Fukuda, R., Tamari, S.I.F., Ming, X., Suzuki, M., 1999. A technique of mathematical expression structure analysis for the hand-writing input system. In: Proceedings of the Fifth International Conference on Document Analysis and Recognition, pp. 131–134.

Garain, U., Chaudhuri, B.B., 2004. Recognition of online handwritten mathematical expressions. IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics 34 (6), 2366–2376.

Grbavec, A., Blostein, D., 1995. Mathematics recognition using graph rewriting. In: Proceedings of the Third International Conference on Document Analysis and Recognition, pp. 417–421.

Kacem, A., Belaïd, A., Ahmed, M.B., 2001. Automatic extraction of printed mathematical formulas using fuzzy logic and propagation of context. International Journal of Document Analysis and Recognition 4 (2), 97–108.

Kanahori, T., Suzuki, M., 2001. A recognition method of matrices by using variable block pattern elements generating rectangular area. In: Proceedings of the Fourth International IAPR Workshop Graphics Recognition, pp. 455–469.

Lamport, L., 1999. LaTeX: A Document Preparation System. Addison-Wesley, Reading, MA.

Miller, E.G., Viola, P.A., 1998. Ambiguity and constraint in mathematical expression recognition. In: Proceedings of the 15th National Conference on Artificial Intelligence, pp. 784–791.

Munich, M.E., Perona, P., 1996. Visual input for pen-based computers. Proceedings of the International Conference on Pattern Recognition, vol. 3, pp. 33–37.

Okamoto, M., Miao, B., 1991. Recognition of mathematical expressions by using the layout structures of symbols. Proceedings of the First International Conference on Document Analysis and Recognition 1, 242–250.

Özer, Ö.F., Özün, O., Tüzel, C.Ö., Atalay, V., Çetin, A.E., 2001. Vision-based single-stroke character recognition for wearable computing. IEEE Intelligent Systems 16 (3), 33–37.

Phillips, I., Chhabra, A., 1999. Empirical performance evaluation of graphics recognition systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 21 (9), 849–870.

Plamondon, R., Srihari, S.N., 2000. On-Line and off-line handwriting recognition: a comprehensive survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (1), 63–84.

Sakamoto, Y., Xie, M., Fukuda, R., Suzuki, M., 1998. On-line recognition of handwriting mathematical expression via network. Proceedings of the Third Asian Technology Conference on Mathematics (ATCM), Tsukuba, Japan.

Smithies, S., Novins, K., Arvo, J., 1999. A handwriting-based equation, editor. In: Proceedings of Graphics Interface, pp. 84–91.

Tang, X., Lin, F., 2002. Video-based handwritten character recognition. Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP'02), vol. 4, pp. 3748–3751.

Toyozumi, K., Suzuki, T., Mori, K., Suenaga, Y., 2001. A system for real-time recognition of handwritten mathematical formulas. Proceedings of International Conference on Document Analysis Recognition (ICDAR), Seattle, WA, pp. 1059–1063.

Wang, Z.X., Faure, C., 1988. Structural analysis of handwritten mathematical expressions. In: Proceedings of the Ninth International Conference on Pattern Recognition, pp. 32–34.

Wienecke, M., Fink, G.A., Sagerer, G., 2001. Video-based on-line handwriting recognition. In: Proceedings of International Conference on Document Analysis and Recognition, pp. 226–230.

Winkler, H., Fahrner, H., Lang, M., 1995. A soft decision approach for structural analysis of handwritten mathematical expressions. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing, pp. 2459–2462.

Zanibbi, R., 2000. Recognition of mathematics notation via computer using baseline structure. Technical Report, School of Computing, Queen's University.

Zanibbi, R., Blostein, D., Cordy, J.R., 2002. Recognizing mathematical expressions using tree transformation. IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (11), 1455–1467.