IJDAR (1999) 2: 37–44

International Journal on Document Analysis and Recognition

An architecture for handwritten text recognition systems

Gyeonghwan Kim[1], **Venu Govindaraju**[2], **Sargur N. Srihari**[2]

[1] Department of Electronic Engineering, Sogang University, CPO Box 1142, Seoul 100-611, Korea; e-mail: gkim@ccs.sogang.ac.kr
[2] CEDAR, State University of New York at Buffalo, 520 Lee Entrance, Amherst, NY 14228–2567, USA

Received October 30, 1998 / Revised January 15, 1999

Abstract. This paper presents an end-to-end system for reading handwritten page images. Five functional modules included in the system are introduced in this paper: (i) pre-processing, which concerns introducing an image representation for easy manipulation of large page images and image handling procedures using the image representation; (ii) line separation, concerning text line detection and extracting images of lines of text from a page image; (iii) word segmentation, which concerns locating word gaps and isolating words from a line of text image obtained efficiently and in an intelligent manner; (iv) word recognition, concerning handwritten word recognition algorithms; and (v) linguistic post-processing, which concerns the use of linguistic constraints to intelligently parse and recognize text. Key ideas employed in each functional module, which have been developed for dealing with the diversity of handwriting in its various aspects with a goal of system reliability and robustness, are described in this paper. Preliminary experiments show promising results in terms of speed and accuracy.

Key words: Handwriting recognition – Page image processing – Word segmentation – Neural networks

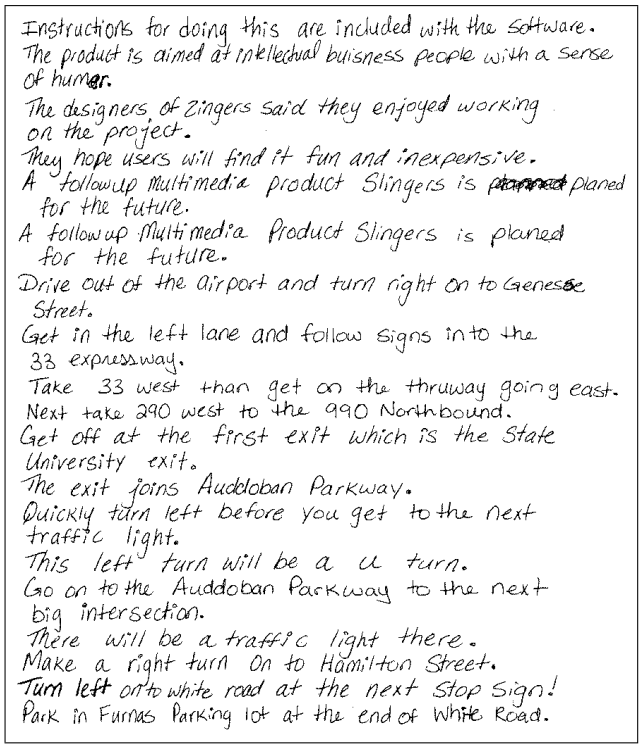

Instructions for doing this are included with the software.
The product is aimed at intellectual buisness people with a sense of humor.
The designers of Zingers said they enjoyed working on the project.
They hope users will find it fun and inexpensive.
A followup multimedia product Slingers is planed for the future.
A followup Multimedia Product Slingers is planed for the future.
Drive out of the airport and turn right on to Genesee Street.
Get in the left lane and follow signs into the 33 expressway.
Take 33 west than get on the thruway going east.
Next take 290 west to the 990 Northbound.
Get off at the first exit which is the State University exit.
The exit joins Audoban Parkway.
Quickly turn left before you get to the next traffic light.
This left turn will be a u turn.
Go on to the Audoban Parkway to the next big intersection.
There will be a traffic light there.
Make a right turn on to Hamilton Street.
Turn left onto white road at the next Stop Sign!
Park in Furnas Parking lot at the end of White Road.

Fig. 1. Sample input to the system described in this paper: a handwritten page image

1 Introduction

Machine recognition of general handwritten text presents a number of challenges. Most methods focus on isolated units, such as characters and words [2,4–6,15]. Recognition of unconstrained handwritten page images is difficult because of the diversity in writing styles, inconsistent spacing between words and lines, and uncertainty of the number of lines in a page and the number of words in a line.

This paper describes an end-to-end system for reading handwritten page images (Fig. 1). One control strategy for exploiting the interactions between the image-level, word-level, and linguistic-level constraints is illustrated in Fig. 2. Inputs to the system are a handwritten page image, a lexicon for the word recognizer, and a corpus for the linguistic post-processor. Output of the system is the ASCII representation of handwriting. Most of the functionality of the architecture involves techniques with considerable scope for improvement.

Principal elements of this paper follow the stages shown in Fig. 2. *Image analysis* issues include image conversion, line separation, and preprocessing. It deals with efficient manipulation of large size image inputs, text line detection, text line extraction, noise removal, and underline removal. *Word segmentation* concerns the

Correspondence to: G. Kim

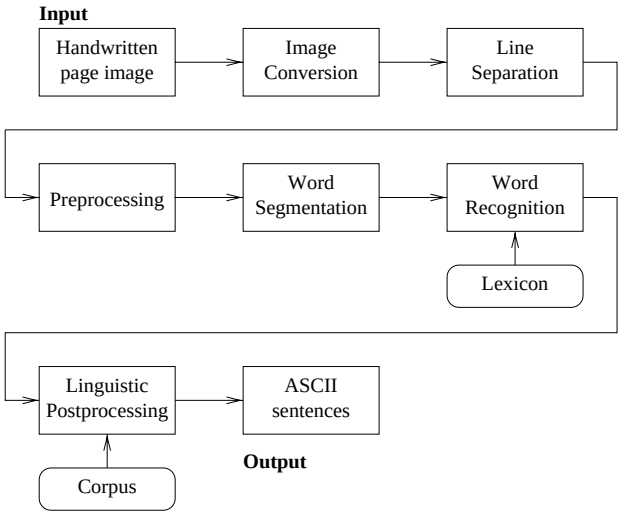


Fig. 2. Overall flow of the system

isolation of constituent words from a line of text image, without help from the recognition module. *Word recognition* concerns handwritten word recognition algorithms. *Linguistic post-processing* concerns the use of linguistic constraints to intelligently parse and recognize text. The recognition phase and the linguistic phase can be intertwined in order to maximally utilize linguistic constraints.

This paper is organized as follows: Sect. 2 explains details of each functional module defined in the Sect. 1. Section 3 briefly describes the graphic user interface of the system. Section 4 is about preliminary experimental results and Sect. 5 gives a summary of the paper and future work.

2 Functional modules

This section describes details of the functional modules shown in Fig. 2.

2.1 Image conversion

US letter size (8.5in × 11in) with 300 dpi (dots per inch) resolution needs storage of 8.4 Mb! An image representation capable of reducing the amount of data to be processed, without loss of information is called for. We use the chain code representation [7]. Functional modules in the system, including line separation, word segmentation and word recognition, work with chain codes.

2.2 Line separation

The task of grouping characters and words into lines of text is relatively straightforward for machine-printed documents. It can be accomplished by examining the horizontal histogram profile at a small range of skew angles [12,17]. The task is more difficult in the handwritten domain. Here, lines of text might undulate up and down,

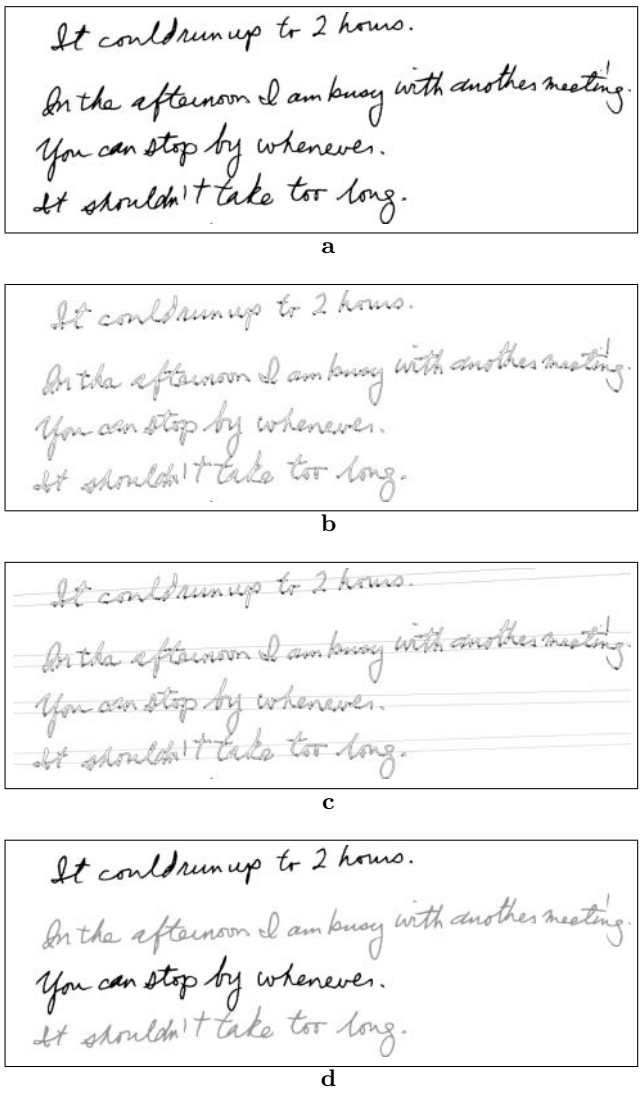


Fig. 3a–d. Line separation procedure - **a** a part of handwritten page image, **b** locating extreme points (dots) on contours, **c** line-groups after clustering, and **d** line segmentation result with different intensity

and ascenders and descenders frequently intersect characters from neighboring lines.

We have developed a method based on the notion that the baseline of a handwritten line is well-defined, as people seem to write on an imaginary line on which the core of each word of the line resides. This imaginary baseline is approximated by the local minima points from each component. A clustering technique is used to group the minima of all the components to identify the different handwritten lines. The following is a brief description of the procedures on the line separation algorithm (Fig. 3).

1 Identify all meaningful components in the image, ignoring components that are likely to be noise or camera artifacts.
2 Find all extreme points (maxima and minima) contributed by each meaningful component. The average

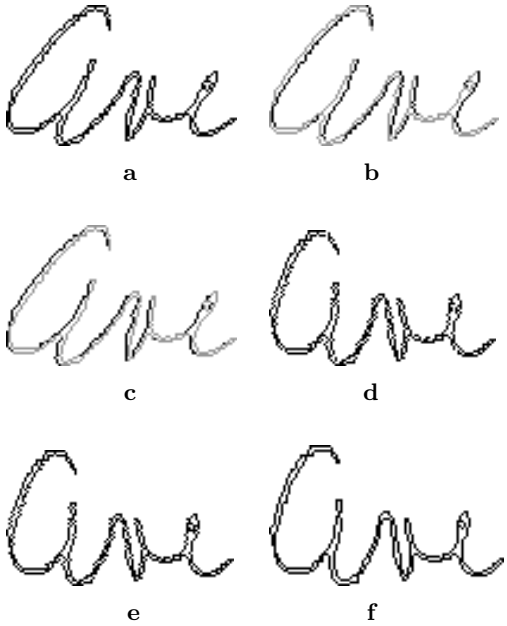


Fig. 4a–f. Pre-processing steps: **a** contour representation of an image, extraction of upward **b** and downward **c** vertical lines, **d** correction of x-coordinates based on the estimated angle, **e** connecting broken points. **f** smoothing contours

y-value difference between consecutive maxima and minima is taken to be the component height (Fig. 3b).

3 Sort all minima by x-value.

4 Using average component height as a useful measure of probable distance between neighboring clusters, begin clustering of the minima from the left side of the image. A new cluster is created when the next minimum is farther than the average component height from all currently existing clusters.

5 Repeat the clustering from the right side of the image.

6 Combine information from both the left and right clusterings. If the left and right clusterings produce the same number of clusters, and membership of each cluster is the same in both the left and right clusterings, the clusters are final. If there is disagreement, a combination strategy is employed.

7 Post-processing is necessary to deal with individual components that contribute minima points to more than one cluster. Decisions must be made to assign components to line-groups based on their contributions of minima points. In some cases, a single component must be forcibly split into two or more components, each of which is then assigned to the appropriate line-group (Fig. 3c).

2.3 Normalization

Noise removal, slant correction and smoothing of chain code contours is performed [8,19].

Noise, which could be introduced during the image capture procedure from scanning devices and transmission media, can be easily identified and compared to the size and shape of connected components (relative to average stroke width).

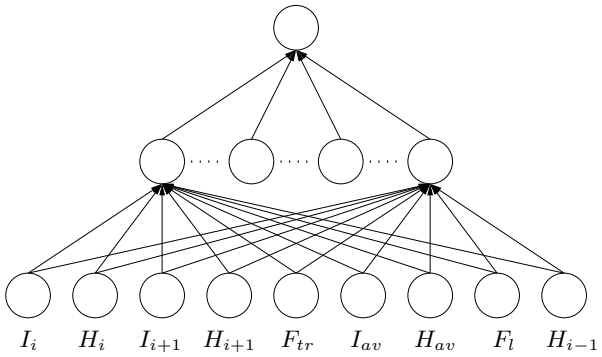


Fig. 5. Structure of the neural network used for the word segmentation: inputs to the network are extracted from character segments

Handwriting samples often have a slant which needs to be corrected for segmentation and recognition (Fig. 4a). Vertical line elements from contours are extracted by tracing chain code components using a pair of one-dimensional filters (Fig. 4b,c). Global slant angle is the average of all angles, weighted by their length in the vertical direction because the longer ones give more accurate information than the shorter ones. Correction is performed based on the estimated slant angle by adjusting x-coordinates of components. Quantization error incurred during the correction introduces "*jigs*" and even broken points (Fig. 4d). The broken points are interconnected by interpolation (Fig. 4e).

The smoothing operation eliminates small blobs on a contour and illegal combination of chain code slopes introduced while the slant correction is being performed. A sliding three-component window is applied. All combinations of three adjacent chain code slopes are analyzed and classified into seven types based on the treatment warranted. Depending on the type, components can be removed from the chain code structure, and coordinates and slopes of components can be updated to maintain the relation between adjacent pixels. For efficient type classification a three-dimensional table is used to store types and corresponding offsets [8]. Figure 4f shows the final pre-processing results after the smoothing operation.

2.4 Word segmentation

After line isolation and extraction, the next challenge is the location of the word boundaries. Since words tend to flow together, it is necessary to develop algorithms to search for the likely boundaries of words. Such techniques are somewhat tied to the word recognition algorithms employed and there are a number of tradeoffs to be considered.

Few approaches in the literature have dealt with the word segmentation issues and most of them have focused on identifying physical gaps using only geometric distances between connected components [13,14,16]. These methods assume that gaps between words are larger than the gaps between characters. However, in handwriting,

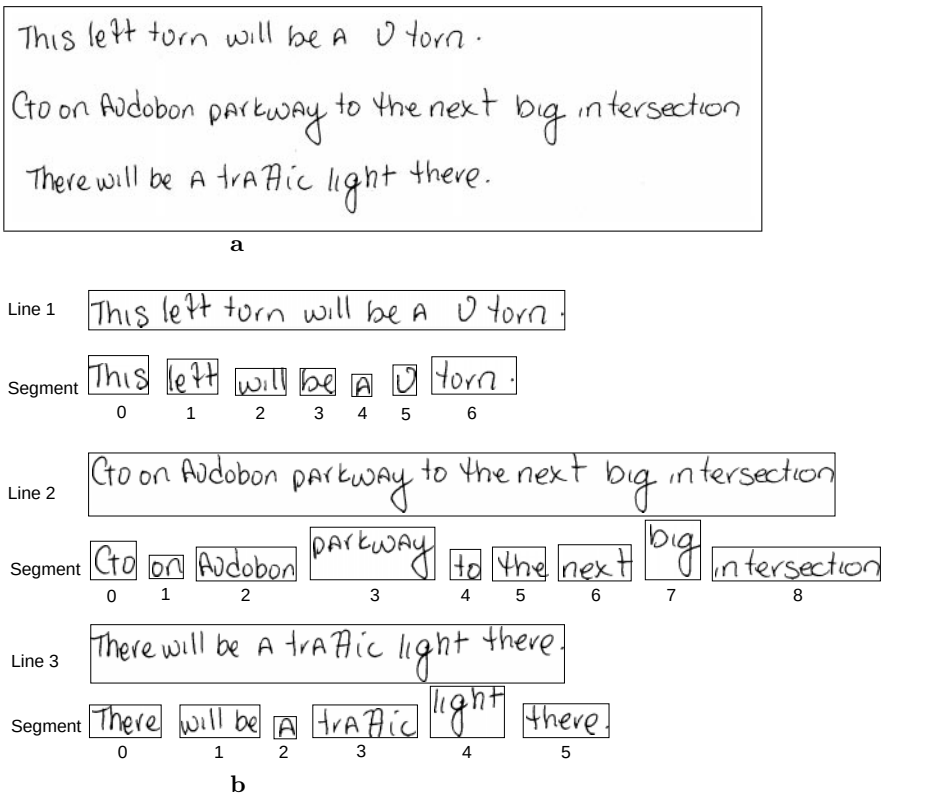


Fig. 6. **a** A sample text image with 3 sentences. **b** Word segmentation results of the new algorithm - most probable word gaps are located using a neural network without any word hypothesis (segment 1 in line 3 is under-segmented). **c** Word segmentation results of several images with different writing styles

exceptions are commonplace because of the presence of flourish in writing styles with leading and trailing ligatures.

We have developed an intelligent word segmentation method that incorporates cues that humans use and does not rely solely on the one-dimensional distance between components [10]. The author's writing style in terms of spacing is captured by characterizing the variation of spacing between adjacent characters as a function of the corresponding characters themselves. The notion of expecting greater space between characters with leading and trailing ligatures is encoded into the segmentation scheme.

A character segmentation algorithm locates segmentation points between characters in a word and is based on the following assumptions: (i) the number of segments per character must be at most 4; and (ii) all touching characters should be separated. Segmentation points are determined using features like ligatures and concavities. Gaps between character segments (a character segment can be a character or a part of character) and heights of character segments are used in the algorithm. In addition, to locate a word break, we use two consecutive intervals and more than two consecutive heights. The character segmentation results are cached for later use in the word recognizer to eliminate duplication of the operation.

A neural network is trained using the information extracted from training images (Fig. 5). The following eight parameters are selected from the character segments for inputs of the neural network: two consecutive intervals (I_i and I_{i+1}) and heights (H_i and H_{i+1}), the

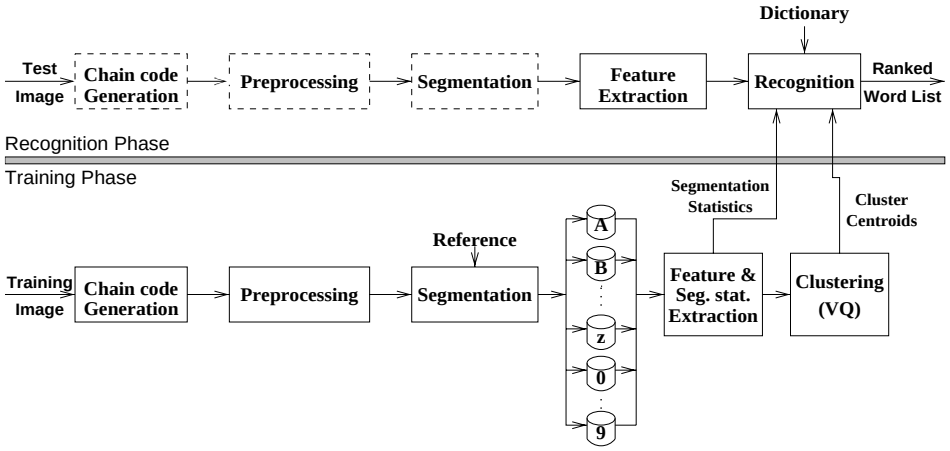


Fig. 7. Overall structure of the word model recognizer - boxes with dashed line are performed in the early stages of the system

average of intervals (I_{av}) and heights (H_{av}), a flag for inter-component transition (F_{tr}), and a flag for the last segment (F_l).

Twenty page images were collected and used for training the neural network. Each line image was fed to the training module and true word gaps were located manually. 18211 training patterns were collected for the purpose.

A sample extracted from a page image is shown in Fig. 6a. As we can see in the figure, locating correct word gaps from line images is a real challenge (without any help from recognition). As shown in Fig. 6b, the word segmentation algorithm we have developed locates the most probable word gaps using the trained neural network without any word hypothesis.

It has been observed that word segmentation is more difficult when the script has discrete writing style (as opposed to cursive). Figure 6c shows word segmentation results on different writing styles. The neural network based method described locates word gaps without any word hypothesis quite accurately for various writing styles. This leads to significant improvement in processing speed as fewer word segments need to be recognized and hence fewer sentences need to be passed to the linguistic post-processor. This becomes significant in light of the fact that the word recognition and the linguistic post-processing modules consume more than 70 percent of the entire processing time (Sect. 4).

2.5 Word recognition

The most important of text recognition subtasks is the development of algorithms for the recognition of (possibly poorly written) isolated handwritten words from a relatively large lexicon of words. There are several recognition algorithms available but none are sufficiently sophisticated for the general text recognition problem. The page image in Fig. 1 shows a sample document whose recognition involves solving the general problem. Specifically, the general problem calls for word recognition with unlimited lexicons (≈50000 words).

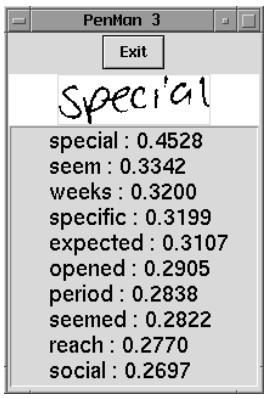
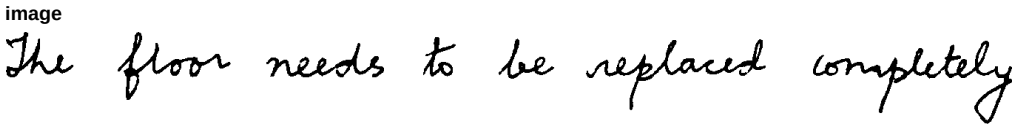


Fig. 8. Results from word recognition showing the confidences of matching

Our word recognition scheme, shown in Fig. 7, uses a matching technique designed to handle the large dimensional feature vectors which represent shape description of characters in a word. The recognizer employs a lexicon-driven approach. The lexicon is introduced in the early stages of the recognition process — an input word image is compared with *only* those words present in the lexicon. In the matching procedure, comparisons between feature vectors of several possible combinations of segments and reference feature vectors of codewords are made to find the best match. For each match, the minimum distance value of each comparison is retained. Also, the concept of variable duration, which is obtained from character segmentation statistics and used for determining the size of the matching window during the recognition is used in the recognizer [11]. The variable duration maximizes efficiency in terms of both speed and accuracy. Figure 8 shows an example of the word recognition result with confidences. Details of the recognizer are described in [9].



word recognition results

the:0.39	from:0.39	need:0.62	**to:0.31**	in:0.52	**replaced:0.52**	**completely:0.47**
she:0.39	**floor:0.38**	**needs:0.56**	tv:0.25	is:0.45	ruined:0.40	butterfly:0.34
like:0.36	for:0.38	field:0.52	is:0.23	on:0.39	walked:0.39	military:0.34
we:0.31	fun:0.35	meet:0.48	as:0.18	**be:0.38**	placed:0.39	immediately:0.33
line:0.31	fear:0.35	area:0.48	ps:0.16	led:0.33	married:0.37	directly:0.33

post-processing results

1. **the floor needs to be replaced completely : -54.328**
2. the fun needs to be replaced completely : -54.367
3. the floor area to be replaced completely : -54.379
4. the floor needs to be replaced directly : -54.484
5. the fun needs to be replaced directly : -54.522

Fig. 9. An example of post-processing that improves sentence level recognition: bold-faced words in the word recognition results show that truth of each word is not always top choice, but the post-processor chose the correct one as the top choice

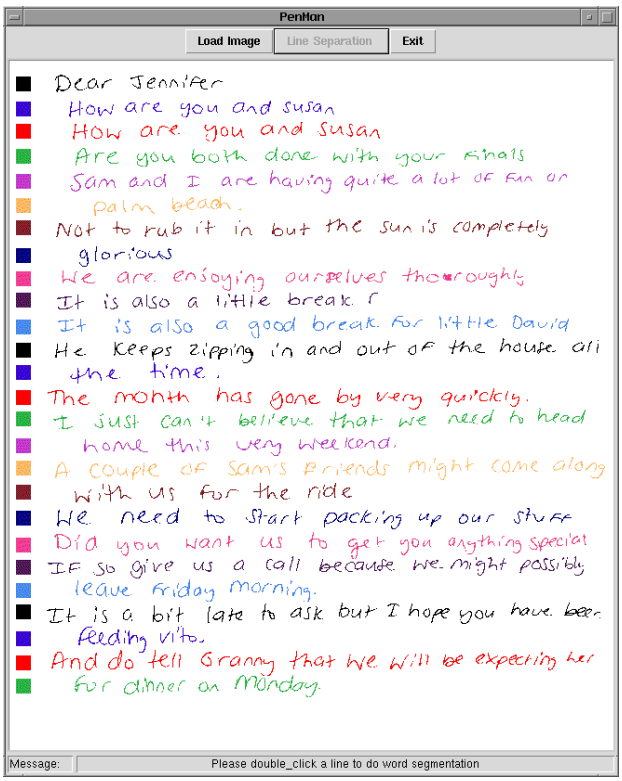


Fig. 10. A typical page image after the line separation: the small square in the beginning of each line represents the line and is used for selecting the line for the subsequent operation

2.6 Linguistic post-processing

Due to variability in handwriting styles and distortions caused by the digitizing process, even the best word recognizers are unreliable when the lexicon size is large. This necessitates the use of linguistic constraints (which employ phrase and sentence level context) to achieve a performance level comparable to that of humans. Input to this stage consists of a sentence/phrase buffer containing the word neighborhoods for each word. As an example, consider Fig. 9. If we restrict the recognizer to return only the top choice for each input word, the sentence will be erroneously read. Although the correct words can be found in the top few choices, it requires the use of contextual constraints in order to override the (sometimes erroneous) top choices. More details on the models employed in the system are described in [1].

Table 1. Timing profile of the system for a typical page image shown in Fig. 10 with 300 dpi resolution (measured on a Sparc 20 platform)

Module	time (sec)	percent
Reading image	1.65	3.0
Loading data(lex, corpus)	8.38	15.3
Chain code generation	1.98	3.6
Pre-processing	0.36	0.7
Line separation	0.40	0.7
Word segmentation	0.85	1.6
Word recognition	23.69	43.3
Ling. Post-processing	17.42	31.8
Total	54.73	100.0

3 Graphic user interface

For evaluation and debugging purposes, we have developed a graphic user interface (GUI) version of the system. We use Tcl/Tk because it is easily interfaced and provides a convenient development environment [3,18]. Each functional module has been modified to produce intermediate files. As the program proceeds by clicking buttons on the screen, the appropriate files are displayed by the tools. A message line is provided at the bottom of each window to guide the operator.

Figure 10 shows results of line separation on a typical page image (Different colors are assigned to the separated lines).

Table 2. Performance - line separation, word segmentation and word recognition

	writing style	discrete	cursive	mixed	total
	# of images	6	5	9	20
Line separation	# of lines	118	101	210	429
	# of lines separated	114 96.6%	97 96.0%	198 94.3%	409 95.3%
Word segmentation	# of words	641	692	1,427	2,760
	# of words segmented	597 93.1%	631 91.2%	1,379 96.6%	2,607 94.5%
Word recognition	top 1	432 72.4%	268 42.5%	750 54.4%	1,450 55.6%
	top 2-10	541 90.6%	459 72.7%	1,081 78.4%	2,081 79.8%

4 Experiments and results

Preliminary experiments have been conducted to evaluate the performance of the system and measure the speed. Performance of the post-processing module was not evaluated because it needs further development.

4.1 Image database

Twenty page images written by different authors and scanned at 300 dpi were used for evaluation. These images are different from ones used for training the neural network (Sect. 2.4).

4.2 Timing analysis

Because each page image contains a different number of lines and words, we have chosen a typical page image with discrete writing style (Fig. 1) for the purpose of timing the system. Table 1 shows the timing profile. It should be noted that the time taken for loading the lexicon and corpus, can be subtracted from the total processing time because these data files are common to all images and do not need to be loaded every time.

The size of the lexicon used for the word recognizer is about 1600 words and the lexicon contains all the words that appear in the image set. The top 5 choices returned by the word recognizer are passed to the post-processing module. If the number of choices is increased, the processing time increases.

The efficiency of the image handling routines is evident from the table as it consumes only 10 percent of the processing time.

4.3 Performance analysis

Table 2 shows performance of line separation, word segmentation, and word recognition for each image across writing styles. The line separation algorithm we developed works equally well over the different writing styles. However, the word segmentation performance varies depending on the writing styles, which suggests more active encoding of the writing style of the author in terms of spacing. A failure case with discrete writing style can be found in Fig. 6. Based on the word recognition performance, the word recognition scheme, which has been developed for applications with small size lexicon, needs to be enhanced so that it can handle applications with a large lexicon.

5 Summary and future work

A complete end-to-end system for reading handwritten page images has been implemented. The system is written in C and has about 30000 lines of code. The processing time for a typical page image scanned with 300 dpi is less than a minute on a single Sparc 20 platform.

For further improvement of the system, the following issues are being considered. The version of the line separation algorithm integrated into the system was originally designed for separating short lines. Therefore, problems are encountered when long and skewed lines are processed. A learning algorithm based on the Hough transform should be considered. In the word segmentation algorithm, incorporation of global information from neighboring short line images should be considered. Behavior analysis of the neural network can give clues on which segment needs to be further segmented and which segment needs to be combined. Modification of the current character segmentation algorithm so that it can be shared by the word segmentation and possibly the line separation algorithms is being considered. Ways to handle punctuation marks should be researched, and the post-processor has to be further improved.

References

1. D. Bouchaffra, E. Koontz, V. Kripasundar, and R. Srihari. Incorporating Diverse Information Sources in Handwriting Recognition Postprocessing. International Journal of Imaging Systems and Technology (IJIST), 7, 1996
2. R. M. Bozinovic and S. N. Srihari. Off-line Cursive Script Word Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(1):68–83, January 1989
3. C. Cracknell and A. C. Downton. A Handwritten Form Reader Architecture. In Proc. of Sixth International Workshop on Frontiers in Handwriting Recognition (IWFHR VI), Korea, pages 67–76, August 1998
4. G. Dzuba, A. Filatov, D. Gershuny, and I. Kil. Handwritten Word Recognition - The Approach Proved by Practice. In Proc. of Sixth International Workshop on Frontiers in Handwriting Recognition (IWFHR VI), Korea, pages 99–111, August 1998
5. C. Farouz, M. Gilloux, and J.-M. Bertille. Handwritten Word Recognition with Contextual Hidden Markov Models. In Proc. of Sixth International Workshop on Frontiers in Handwriting Recognition (IWFHR VI), Korea, pages 133–142, August 1998
6. J. T. Favata, G. Srikantan, and S. N. Srihari. Handprinted Character/Digit Recognition using a Multiple Feature/Resolution Philosophy. In The fourth International Workshop on Frontiers in Handwriting Recognition, Taipei, Taiwan, pages 57–66, 1994
7. H. Freeman. Computer Processing of Line-Drawing Images. Computing Surveys, 6(1):57–97, 1974
8. G. Kim and V. Govindaraju. Efficient Chain Code Based Image Manipulation for Handwritten Word Recognition. In Proc. of the SPIE symposium on electronic imaging science and technology (Document Recognition III), San Jose, CA, volume 2660, pages 262–272, February 1996
9. G. Kim and V. Govindaraju. A Lexicon Driven Approach to Handwritten Word Recognition for Real-Time Applications. IEEE Transactions on Pattern Analysis and Machine Intelligence, 19(4):366–379, April 1997
10. G. Kim and V. Govindaraju. Handwritten Phrase Recognition as Applied to Street Name Images. Pattern Recognition, 31(1):41–51, 1998
11. G. Kim, V. Govindaraju, and S. N. Srihari. Handwritten Word Recognition using Dynamic Matching with Variable Duration. In Proc. of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP96), Atlanta, volume 6, pages 3454–3457, May 1996
12. L. Likforman-Sulem, A. Hanimyan, and C. Faure. A Hough Based Algorithm for Extracting Text Lines in Handwritten Documents. In Third International Conference on Document Analysis and Recognition (ICDAR-95), Montreal, Canada, pages 774–777, 1995
13. U. Mahadevan and R. C. Nagabushnam. Gap Metrics for Word Separation in Handwritten Lines. In Third International Conference on Document Analysis and Recognition (ICDAR-95), Montreal, Canada, pages 124–127, 1995
14. G. Seni and E. Cohen. External Word Segmentation of Off-Line Handwritten Text Lines. Pattern Recognition, 27(1):41–52, 1994
15. J. C. Simon. Off-line Cursive Word Recognition. Proceedings of the IEEE, 80(7):1150–1161, 1992
16. S. Srihari, V. Govindaraju, and J. Favata. Unconstrained Handwritten Text Recognition. In Proceedings of 1995 Symposium on Document Image Understanding Technology, Bowie, Maryland, pages 226–235, 1995
17. S. N. Srihari and V. Govindaraju. Analysis of Textual Images Using the Hough Transform. Machine Vision and Applications, 2:141–153, 1989
18. B. B. Welch. Practical Programming in Tcl and Tk (2nd ed.). Prentice Hall, 1997
19. D. Yu and H. Yan. An Efficient Algorithm for Smoothing, Linearization and Detection of Structural Feature Points of Binary Image Contours. Pattern Recognition, 30(1):57–69, 1997

Gyeonghwan Kim received the B.S. (cum laude) and the M.S. in electronic engineering from the Sogang University, Korea, in 1984 and 1986, respectively. He received the Ph.D. degree in electrical and computer engineering from the State University of New York at Buffalo, in 1996. From 1986 to 1991 he was a researcher in the Goldstar (currently LG) Precision Technology Institute (Korea). From 1993 to 1997, he was with the Center of Excellence for Document Analysis and Recognition(CEDAR) at SUNY at Buffalo as a research scientist. He is currently an Assistant Professor in the Department of Electronic Engineering at the Sogang University, Seoul Korea. His research interests include in the areas of handwriting recognition, document image analysis, neural networks, and human-computer interface.

Venu Govindaraju is the Associate Director of the Center of Excellence for Document Analysis and Recognition at SUNY Buffalo. He is the project manager and senior research scientist for the CEDAR Handwritten Address Interpretation Project. He holds the Research Associate Professorship in the Department of Computer Science at SUNY Buffalo. Govindaraju received his BTech (Hons.) in Computer Science from the Indian Institute of Technology at Kharagpur in 1986, and his MS and PhD in Computer Science from SUNY Buffalo in 1988 and 1992 respectively. Dr. Govindaraju has co-authored more than 70 research papers in various conferences and journals in the fields pattern recognition and computer vision. Dr. Govindaraju is currently the associate editor of The Pattern Recognition Journal and Chair of the IEEE SMC subcommittee on Pattern Recognition.

Sargur N. Srihari is the founding director of the Center of Excellence for Document Analysis and Recognition (CEDAR). He has been a faculty member in the Computer Science Department at the State University of New York at Buffalo since 1978, where he is presently a SUNY Distinguished Professor. Srihari received a B.Sc. in Physics and Mathematics from the Bangalore University in 1967, a B.E. in Electrical Communication Engineering from the Indian Institute of Science, Bangalore in 1970, and a Ph.D. in Computer and Information Science from the Ohio State University, Columbus in 1976. He received a New York State/United University Professions Excellence Award for 1991. He became a Fellow of the Institute of Electronics and Telecommunications Engineers (IETE, India) in 1992, a Fellow of the Institute of Electrical and Electronics Engineers (IEEE) in 1995, and a Fellow of the International Association for Pattern Recognition in 1996.