A GESTURE-BASED EDITOR FOR SHORT HANDWRITTEN MESSAGES

Jean-Jules Brault, Réjean Plamondon and André Laframboise

Laboratoire Scribens
École Polytechnique de Montréal
P.O. Box 6079, Downtown Station
Montréal, Québec, Canada, H3C 3A7
e-mail: brault@scribens.polymtl.ca
26 mars 1995

Abstract

The messages that we send are often short enough to fit on one or two pages. Because these memos are not intended to be integrated into another document, it is not necessary to edit them with an ASCII word processor or sophisticated graphics software. This project describes a working prototype of a handwritten entity editor operating on an electronic slate and using gestural commands that simulate the paper/pen/scissors/glue paradigm. The system analyzes the sampled handwritten coordinates in real time, and segments them following a hierarchical structure allowing, for example, the "wordwrap" feature of conventional wordprocessing. The segmentation and recognition of the gestural commands of the system are evaluated on sixteen users.

1. Introduction

The arrival on the market of electronic slates (note-pads) has opened up new areas for research and applications. This new type of computer uses the display screen as a writing surface, and generally consists of a transparent digitizing tablet superimposed on a liquid crystal display screen. The pen generates a trace on the screen (called "electronic ink") which can be erased, moved, copied and modified at will by the user. The screen becomes an "intelligent" sheet of paper, thus enabling the development of a very wide variety of new applications. In addition, notepads now integrate numerous functions usually handled by separate systems like fax machines, e-mail, cellular phones and pagers. These new, integrated systems are known as personal digital assistants (PDAs) or personal communicators (PCs).

The first electronic slates to be commercially successful were the GRiDPADs, marketed by GRiD Systems in 1989. Note that a number of research laboratories [1,3,7,8,11,13,16], have developed prototypes enabling study of the equipment and software aspects of this new type of tool. Today, several major companies offer similar products (Apple, Compaq, EO, Fujitsu, IBM, NCR, NEC, Sharp, etc.) and operating systems have been designed to enable the development and use of the pen-screen interface for many existing applications, as well as future ones (Microsoft's *Windows for Pen*, PenDos, PenRight!, GEOS 2.0 SDK, Newton Intelligence). The reader wishing a more complete picture of who is involved in this emerging industry (called "nomad computing") and information on product suppliers can consult specialized publications geared to this market (like PEN Magazine [14] and Pen Computing Magazine [15]).

Although the equipment is improving rapidly, there are still major steps to be taken in terms of sampling and display resolution, and in computing power. However, it is in the software domain that improvements are most eagerly awaited. In fact, the two most important problems that will have to be resolved are the recognition of handwriting and the recognition of the subjacent structures of a document.

Handwriting recognition involves not only alphanumeric characters and cursive writing (i.e. nonsegmented), but also the various symbols and drawings encountered in other spheres of activity.

Structure recognition, on the other hand, applies not only to textual structures (lines, paragraphs, columns, etc.), but also to drawing structures (like the topologies of electronic circuits). This is a much younger research domain than that of isolated character recognition, however. It was often neglected because their subjacent structure being more or less taken for granted. For the past few years, it has been the subject of serious study, particularly in the area of printed documents [2,9,10]

Some electronic slate applications do not require the recognition of alphanumeric characters. This is the case, for example, in editing handwritten text. A complete, effective and ergonomic handwritten text editor would be a powerful and useful tool for application where a printed text is not a requirement. The idea of an handwritten text editor is not new. As long ago as 1969, Coleman [4] suggested the use of gestural commands for editing printed text on the screen. Since then, a number of studies have been carried out in this domain [5,7,12,17] to mention a few.

The object of this paper is to report on the steps we have taken in building a prototype of editor for handwritten entities. This system is designed for use in the fairly specific but frequently encountered case where the user must quickly write, edit and send short and clear messages consisting of one or two pages of text and simple drawings.

The article is structured as follows: We analyze the needs of the users of paper notepads, which we all are, in order to determine the needs of users of electronic ink editors. The implementation of tools that can respond to these needs is then described, with an emphasis on the capabilities that such a system should possess, namely stroke interpretation, and the segmentation and structuring of handwritten entities following execution of a command. The results of an evaluation carried out among sixteen writers, with comments justifying the suggested guidelines is given in the conclusion.

2. The needs of paper notepad users

Writing involves four operations which are carried out on a cyclical basis: thinking, writing, reading and modifying (or editing) what has been written. All the modifications that the writer can make to his text are be made with the following two activities: *inserting (adding)* and *deleting (removing)*. The basic tools that he needs are something to write on (paper), something to write with (a pen or pencil) and a means for erasing (for example, an eraser). Of course, how effectively the writer carries out this sequence of operations depends on his or her experience. Thus, the addition of a third editing action, for example *move*, will make the writer's use of the scissors/glue combination (the so-called "cut-and-paste" technique) more efficient. The "paper/pen/eraser/scissors/glue" paradigm has serious limitations, however, because once the writer stops "cleaning" the text, homogeneity and legibility degrade rapidly.

3. The needs of electronic notepad users

Although an electronic notepad is a computer, the minimal needs of a user of an electronic ink editor for preparing and transmitting short handwritten messages (text and drawings) are different from those of the user of a conventional word processor (or graphics software) for preparing a thesis, for example. Our objective here is to exploit and improve the skills that all writers have been able to develop with the "paper/pen/eraser/scissors/glue" paradigm, without too much modification of their skills and expectations. The same tools for editing text on paper notepads must be reproduced for the electronic ink editor. However, it would be easy to envisage adding other functions, given that electronic ink is virtual and held in memory (i.e. *carriage return* used to create a new paragraph, and *printing*, *faxing* and *copying*).

One of the principal differences between the "paper/pen..." paradigm and its virtual counterpart is that editing activities must be executed by the system and not by the writer. Because the commands have to be interpreted, errors may be introduced: errors of interpretation! These errors are of two types. The first are related with the recognition of gestural commands as such, and the second, with the identification of the entity on which the command is to be executed. An unequivocal means for identifying this entity (text or drawing) must be found, and for this, a new activity, *select*, needs to be added to the editor. Note that this command must also be recognized...

The correct execution of several commands is based on the assumption that handwritten entities are organized into an adequate data structure, a constraint which is subjacent to the "reorganization" capability of the electronic notepad. Indeed, when a subject writes a series of words, he leaves a space between them to make the words easier to read. However, since we frequently lift the pen within a word as well as at the end of it, thereby creating supplementary spatial discontinuities in the words, the software must continually interpret any discontinuities as either intra-word or inter-word. Additional functions must be added to the editor, therefore, for cases where the spacing has been misinterpreted. These functions are: *link* (...of parts of a word mistakenly segmented), *unlink* (...of a group of words mistakenly linked together), and ultimatly "*undo*".

4. Implementation

Development was carried out on the Microsoft operating system *Windows for Pen*. Based on the needs of users of electronic notepads, the capabilities of the editor are grouped here for discussion under six themes: 1. Coordinates acquisition, 2. Data discrimination between gestural commands or handwritten components, 3. Segmentation and structuration (...of handwritten components), 4. Commands recognition. 5. Modification of the manuscript displayed, 6. Personalization of thresholds. These themes constitute the principal modules of the system that have been developed [12].

4.1 Acquisition and segmentation of coordinates

The IBM-compatible GRiD 2260 was used for data acquisition, which consists of a transparent digitizing tablet superimposed on a liquid crystal display screen The coordinates (x(t), y(t)) are recorded sequentially while the tip of the pen is in contact with the surface ofthe tablet. A series of points, called here after a component, start with a pen-down movement and end with a pen-up movement. This constitutes the first level of segmentation. The second level, the fusion of components, is reached when they are grouped into blocks (groups of spatially close components) which will subsequently be processed as words, graphic elements or handwritten commands. This fusion process is governed by a threshold that can be modified by the user. We decided for a very simple proximity measure, which is the spatial distance as measured along the base line (or line) of writing (note that *the NotePad* application supplied with Microsoft's *Windows for Pen* uses time as a segmentation parameter which has been found inadequate and frequently misleading in our application.

4.2 Discrimation between gestural commands and handwritten components

The question here is, do we have to spell out to the system the nature of future strokes or let the system guess what they will be? We have decided to implement all the possibilities with the aid of a "pencil case" metaphor integrated in a tools bar at the top of the writer's work space (Figure 1). For this project, five tools were defined, corresponding roughly to elements usually found in a conventional pencil case, namely a lead pencil (to enter only handwritten text), a blue pencil (to enter both handwritten text and gestural command), a red pencil (to enter only gestural commands), a drawing pencil (to enter only drawing) and an eraser. The use of the blue pencil requires an algorithm to determine if the blocks of components obtained after the second level of processing (done by the acquisition module) are handwritten components or gestural commands. This algorithm performs rapidly since the gestural commands (shown in Table 1) are quite different from handwritten text. If three or more of the five following tests are not fulfilled, the algorithm classify the block as a command: (1) the number of handwritten components must be higher than three, (2 and 3) the number of maxima and minima (according to the y axis) must be higher than two, (4) the aspect ratio of the block must be higher than a threshold, and (5), the intersecting surface of componants must be lesser than a threshold.

The list of "text" blocks is then sent to the post-segmentation and structuration module (section 4.3), and the list of "gestural command" blocks is sent to the editing command recognition module (section 4.4).

4.3 Post-segmentation and structuration

The system must be able to recognize a minimum number of handwritten entities (the word, line and paragraph for the text, and, elements and figures for the drawings). The type of manuscript recognized by our system is made up of a sequence of pages, each one composed of a top/down sequence of paragraphs (and figures, if there are any) laid on one column. Each paragraph is composed of a top/down sequence of lines, themselves composed of a left/right sequence of words. Each figure is composed of a variety of graphic elements.

The blocks that were written in an empty space on the page are arranged, one after another, on one or more lines. This process is governed by user-modifiable fusion thresholds, which make it possible to decide where the blocks belong. Those written above the blocks already identified as text are fused (accents which will be added later, words that have "i"s to be dotted and "t"s to be crossed, etc.)

4.4 Recognition of editing commands

Table 1 shows the 11 gestural commands implemented as well as their meaning (the number identify each command for futur references as in Figure 4d). Again, the range of possible gestures is very broad. However, we wanted them to be simple, coherent, easy to remember and to recognize.

The input of this module is the list of "gestural command" blocks (see section 4.2). This list is examined in order to group, or split up, the components of each blocks according to a spatial threshold. That can be modified anytime by the user (see section 4.6). Each command to be recognized is

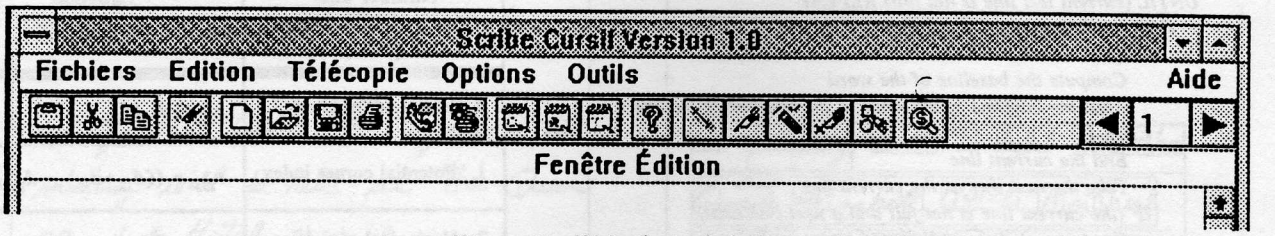

Figure 1: Writer's tools bar.

Commands and Gestures	Commands and Gestures	Commands and Gestures	Commands and Gestures
SELECT	ERASE	COPY	MOVE
INSERT	LINK	UNLINK	RETURN

Table 1: Gesture commands. The number identify each command for futur references as in Figure 4d.

composed of one or more components. The recognition process is carried out here in 3 steps: (1) Extraction of five representative points for each component; (2) Calculation of discriminant measures based on these five points; (3) Recognition of the command using a Bayes classifier.

The representative points of a component are, besides the first and last points of the component, those located at the 1/4, the 1/2 and the 3/4 of the time taken to execute the component. These five points will be referred to as points 1 to 5. The calculation of the 12 discriminant measures (defined in Table 2) is achieved with the aid of these representative points. The parameters used for each measure is shown in Figure 2.

The recognition of a command is the last step in the process. It is based on a two-by-two comparison of specific subsets of discriminant measures (of Table 2) for each of the 36 pairs ((8+1)8/2) of different commands. Note that there are nine different building strokes for the ten gesture commands. For example, the subset used to discriminate the building stroke "opening crochet" from the "arrow tip" was found to be 5, 6 and 7. A two classes Bayesian classifier is used to discriminate each pair of classes. The one that wins more often must also wins a minimum number of time. If the threshold is not passed, the command is said to be "not recognized" and the writer must repeat the gesture command.

4.5 Modification of the document to be displayed

The most interesting aspect, but also the most problematic, one of the modification module is, of course, the reorganization of the document following the execution of a command. It is through this process, for example, that empty spaces resulting from the deletion of words are eliminated. The success of the reorganization mechanisms lies in the quality and accuracy of the segmentation, and in the structure of the subjacent data. After an execution of one of the commands 5, 6, 7, 9, 10, or 11, each line of a given paragraph is modified according to the "reentrant" algorithm described by this pseudocode:

Compute the maximum length of a line
UNTIL (current text line is not full) REPEAT
FOR (each concerned word of the line)
IF (the current word is within the current line)
Compute the baseline of the word
Ajust the baseline of the word to the baseline of the line
ELSE IF
End the current line
Take the next line as the current line
IF (the current line is not full and a next line exist)
Concatenate the next line to the current line
Reorganize the current line

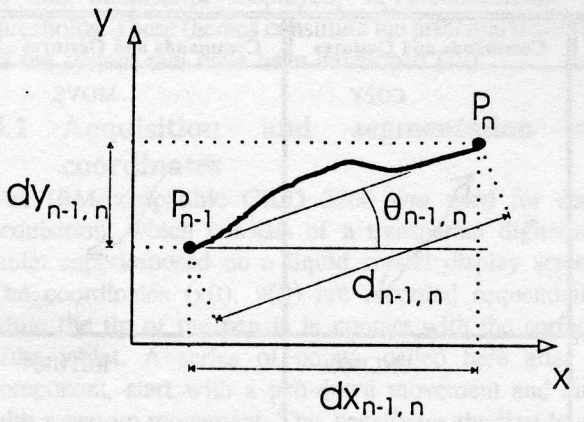

Figure 2: Parameters θ, d, d_x and d_y computed from two representative points P_{n-1} and P_n of a component (defined here as an handwritten trace between two consecutive pen-lifts).

4.6 Personalizing thresholds

The system must be able to adapt its segmentation and recognition parameters to the handwriting and particular preferences of individual users. To mention a few, the users are able to modify the different spacings between words, lines and paragraphs, before and after reorganisation. He also have a choice of recognizers.

5. Example of operation

Figure 3a shows a short handwritten message (with text and drawing) as entered initially. Figure 3b shows the same message as segmented by the system with boxes around the words, lines, paragraph and elements of the drawing in order to show to the user the handwritten entities as part of a hierarchical structure. We note however that some text segmentation errors have occured. Figure 3c shows the gestural commands to correct these errors as well as other commands to modify the original text. The figure shows these commands as entered by the user. It should be noted that it is not possible to see Figure 3c on the user screen since the commands are actually executed as soon as they are recognized by the system. Figure 3d shows the final result of the text part of the message.

Number and Name of the measure (ratio)	Equation
0. Relative dimension	$r_{15} = d_{15}/d_{max}$
1. Potential corner index	$r_{135} = ((d_{13}+d_{35}) - d_{15})/d_{15}$
2. Horizontal projection	$r_{x_{15}} = d_{x_{15}}/d_{15}$
3. Vertical projection	$r_{y_{15}} = d_{y_{15}}/d_{15}$
4. Signed local curvature (at middle time point)	$r_{\theta_{135}} = ((\pi - \theta_{13}) + \theta_{35})/\pi$
5. Signed local curvature (at ¼ time point)	$r_{\theta_{123}} = ((\pi - \theta_{12}) + \theta_{23})/\frac{\pi}{2}$
6. Signed local curvature (at ¾ time point)	$r_{\theta_{345}} = ((\pi - \theta_{34}) + \theta_{45})/\frac{\pi}{2}$
7. Curvilinear	$r_{\Sigma d} = \frac{(d_{12}+d_{23}+d_{34}+d_{45})}{(d_{13}+d_{35})}$
8. Partial horizontal projection (first quarter)	$r_{x_{12}} = d_{x_{12}}/d_{12}$
9. Partial horizontal projection (last quarter)	$r_{x_{45}} = d_{x_{45}}/d_{45}$
10. Partial vertical projection (middle part)	$r_{y_{24}} = d_{y_{24}}/d_{24}$
11. Curvature sign dominancy (for measures 4,5,6)	$S = sign(r_{\theta_{135}}) + sign(r_{\theta_{123}}) + sign(r_{\theta_{345}})$

Table 2: Definitions and symbols of the 12 measure ratios used to discriminate between the 9 building strokes of gesture commands.

6. Preliminary evaluation

To evaluate the system, we carried out two experiments to test the following separately: (1) the segmentation of text based on textual structure; (2) the recognition of gestural commands. These experiments were conducted with the help of two groups of 16 individuals.

For the first experiment (on segmentation), one group used the cursive editor. They had quite varied writing styles which means that segmentation could be tested in a number of different situations. We asked them to transcribe, in cursive, two typed

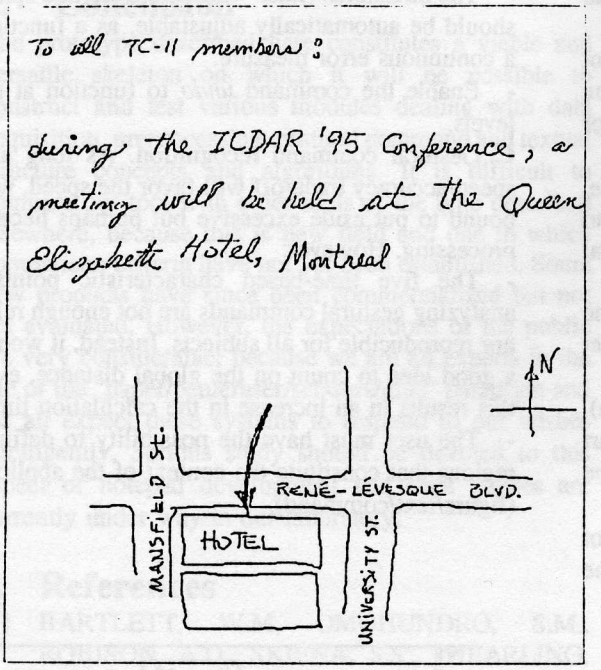

Figure 3a

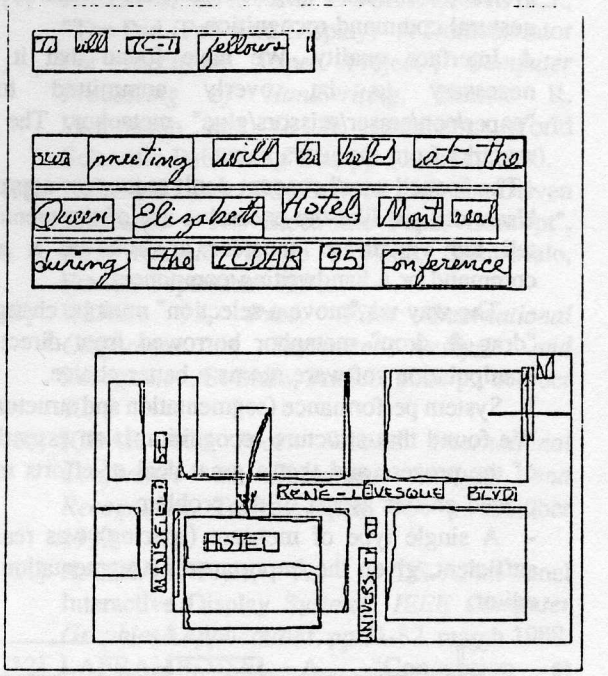

Figure 3b

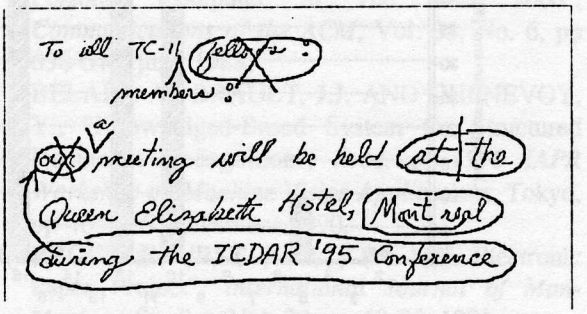

Figure 3c

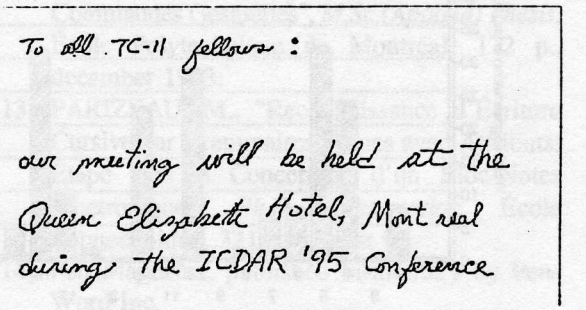

Figure 3d

Figure 3: Example (see text for details).

paragraphs of 3 lines each (equivalent to 5 to 8 handwritten lines). The second group of individuals was asked to read these handwritten texts and to segment them into words, lines and paragraphs. This enabled us to compare the segmentation performed by the software with that carried out by the human subjects.

The percentages of error obtained were found to be highly dependent on the writer. The segmentation into words (Figure 4a) of the writing of seven of the writers was performed with less than 5% error. However, error rates as high as 30% were also obtained. On average, the mean error rate for the segmentation into words was 11 %. The results were, of course, better for the segmentation into lines (Figure 4b), with a mean error rate of 8.5 %). It appeared that the same writers were responsible for the poor performances of the system in both cases. Samples from these writers will be shown at the conference.

For the second experiment, the recognition of gestural commands (see Table 1 for the numbering of the gestural commands) was carried out on a blank, text-free, page. Figure 4c shows the error rates obtained as a function of the writer and Figure 4d shows the error rates obtained for each of the gestural commands. A mean recognition rate of 85% was obtained.

7. Guidelines

Prototyping a complete system has lead to specify more clearly some of the requirements that will have to be met in a next version. These recommendations are grouped under three headings: interface quality,

system performance (segmentation and structuration), and gestural command recognition.

A. Interface quality. We have found that it is not necessary to be overly committed to the "paper/pen/eraser/scissors/glue" metaphor. The major remarks from the users were:

- The "pencil case" concept don't seem very appropriate. Users did not like the excessive use of the menu bar to tell the system to consider a stroke as a gestural command or a handwriting component.
- The way we "move a selection" must be change. The "drag & drop" metaphor borrowed from direct-object manipulation software seems a better choice.

B. System performance (segmentation and structuration). We found that structure recognition is an essential part of the process and that a great deal of efforts must be placed on this aspect of the problem.

- A single type of measure (spacing) was really not sufficient, given the importance of segmentation in the editor.
- The thresholds (inter-word and inter-line spacing) should be automatically adjustable, as a function of a continuous error measure.
- Enable the command *undo* to function at many levels.

C. Gestural command recognition. As long as the speed/accuracy trade-off will favor the speed, we are bound to put aside excessive but perhaps necessary processing. However:

- The five time-based characteristic points for analyzing gestural commands are not enough reliable nor reproducible for all subjects. Instead, it would be a good idea to count on the global distance, even if this results in an increase in the calculation time.
- The user must have the possibility to define the regions that constitute the context of the application (figure/text/command).

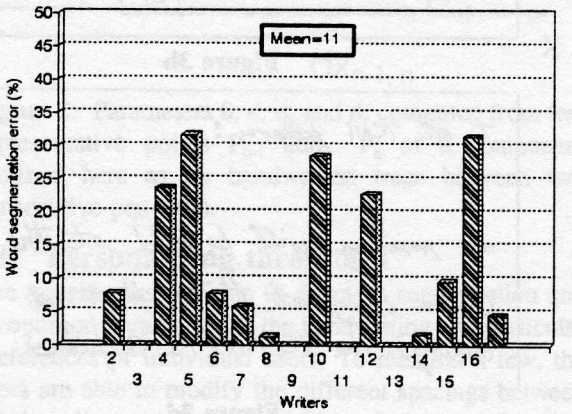

Figure 4a

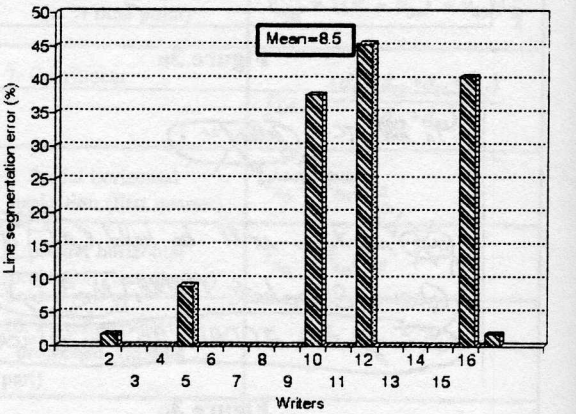

Figure 4b

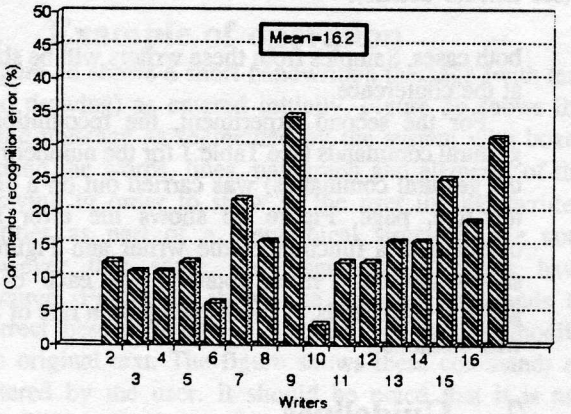

Figure 4c

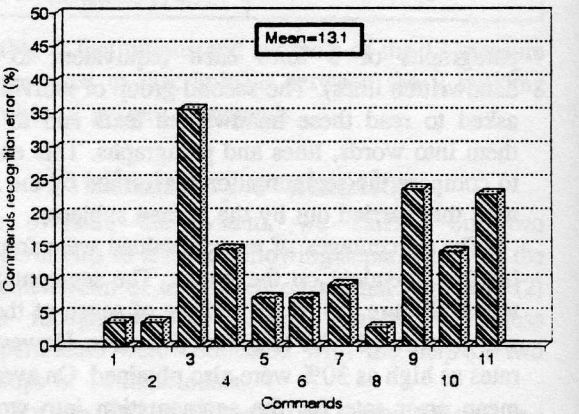

Figure 4d

Figure 4 A. Word segmentation error rate for each of the 16 writers. B. Line segmentation error rate for each of the 16 writers. C. Command recognition error rate for each of the 16 writers. D. Command recognition error rate for each of the 11 commands shown in table 1.

8. Conclusion

The prototype described here constitutes a viable and versatile skeleton on which it will be possible to construct and test various modules dealing with data acquisition, preprocessing, gestural command and textual structure concepts and algorithms. It is difficult to compare our tool with other tools of the type described elsewhere, because this is new field and one in which comparison criteria have not yet been established. Some new products have since been commercialized but not yet evaluated. However, the expectations of the public are very considerable, because we are all experts in the use of the "paper/pen/eraser/scissors/glue" paradigm and we all expect these systems to respond to our wishes intelligently. Serious study should be devoted to this aspect of notepad development and such studies are currently under way in our laboratory.

9. References

[1] BARTLETT, W.M, OMOHUNDRO, S.M., ROBISON, A.D., SKIENA, S.S., THEARLING, K.H., YOUNG, L.T., WOLFRAM, S., "Tablet: Personal Computer In The Year 2000", *Communications of the ACM*, Vol. 31, No. 6, pp 638-646, june 1988.

[2] BELAÏD, A., BRAULT, J.J. AND CHENEVOY, Y., "Knowledged-Based System for Structured Document Recognition", *Proc. of 1990 IAPR Workshop on Machine Vision Applications*, Tokyo, pp 465-469, november 1990.

[3] BROCKLEHURST, E. R., "The NPL Electronic Paper Project", *International Journal of Man-Machine Studies*, Vol. 34, pp 69-95, 1991.

[4] COLEMAN, M.L., "Text Editing on a Graphic Display Device Using Hand-Drawn Proofreader's Symbols", *Pertinent Concepts in Computer Graphics: Proc. Second Univ.of Illinois Conf. Computer Graphics*, M.Faiman and J.Nievergelt, (eds.), Univ. of Illinois Press, Champaign, Ill., pp 282-290, 1969.

[5] DOSTER, W. and OED, R., "Word Processing with On-Line Script Recognition", *IEEE Micro*, pp 36-43, october 1984

[6] HANSEN, C, "Pen Computers in the Federal Government", *PEN*, pp 51-53, march/april 1993.

[7] HIGGINS, C.A. AND DUCKWORTH, R.J., "The Pad (Pen And Display)- A Demonstrator For Electronic Paper Project", *Computer Processing of Handwriting*, Editors R. Plamondon and C. G. Leedham, World Scientific Publishing Co., pp 111-131, 1990.

[8] HIGGINS, C.A., FORD,D. M., "Stylus Driven Interfaces - The Electronic Paper Concept", *First ICDAR 91*, pp 853-862, Saint-Malo, France, September 1991.

[9] ICDAR 91, *Proc. First International Conference on Document Analysis and Recognition*, St-Malo, France, 1010 p., october 1991

[10] ICDAR 93, *Proc. Second International Conference on Document Analysis and Recognition*, Tokyo, Japan, 964 p., october 1993

[11] KANKAANPAA, A., "FIDS-AFlat-Panel Interactive Display System", *IEEE Computer Graphics&Applications*, pp 71-82, march 1988.

[12] LAFRAMBOISE, A., "Conception et Réalisation d'un Éditeur d'Écriture Cursive à Commandes Gestuelles", *M.Sc.(Applied) Thesis*, École Polytechnique de Montréal, 172 p., december 1993.

[13] PARIZEAU, M., "Reconnaissance d'Écriture Cursive par Grammaires Floues avec Attributs: Étape vers la Conception d'un Bloc-Notes Électronique", *Ph.D. Dissertation*, École Polytechnique, 321 p., august 92.

[14] Pen Magazine, published bi-monthly by Pen-Word Inc.

[15] Pen Computing Magazine, published bi-monthly by Pen Computing, Inc.

[16] POBGEE, P.J., "A prototype System for Interactive Input of Cursive Information", *National Physical Laboratory Report DITC 125/88*, september 1988.

[17] WELBOURN, L. K. AND WHITROW, R. J., "A Gesture Based Text And Diagram Editor", *Computer Processing of Handwriting*, R. Plamondon and C. G. Leedham Eds., World Scientific Publishing Co. pp 221-234, 1990.