투명한 입력오버레이를 이용한 필기 및 음성 입력

(Handwriting and Voice Input using Transparent Input Overlay)

김 대 현 † 김 명 준 †† 이 진 호 †††
(Dae-Hyun Kim) (Myoung-Jun Kim) (Zino Lee)

요 약 본 논문은 IBM ViaVoice와 마이크로소프트 필기-인식 시스템(handwriting-recognition system)과 같은 인식 엔진(recognition engines)을 UMPC와 같은 펜-입력 디스플레이에서 작동하는 일반적인 윈도우 어플리케이션과 연동하기 위한 단일화된 멀티모달 입력 프레임워크(unified multi-modal input framework)를 제안한다. 사용자가 펜-입력 디스플레이에 부착되어있는 버튼을 한 손으로 누르면, 인터넷 검색 윈도우나 워드 프로세서와 같이 현재 포커스를 갖는 윈도우는 전체 데스크탑을 덮을 수 있는 투명한 윈도우로 덮여진다. 이 위에 사용자는 다른 한 손으로, 현재 working context를 놓치지 않으면서, 필기 입력을 자유롭게 수행할 수 있다. 이런 투명 입력 윈도우를 이용하여 필기 및 음성 입력뿐만 아니라 다이어그램까지 그릴 수 있게 해준다.

키워드 : 사용자 인터페이스, 입력 프레임워크, 투명 윈도우, 필기체 인식, 음성 인식, 스케치 패드, 사용자 테스트

Abstract This paper proposes a unified multi-modal input framework to interface the recognition engines such as IBM ViaVoice and Microsoft handwriting-recognition system with general window applications, particularly, for pen-input displays. As soon as user pushes a hardware button attached to the pen-input display with one hand, the current window of focus such as a internet search window and a word processor is overlaid with a transparent window covering the whole desktop; upon which user inputs handwriting with the other hand, without losing the focus of attention on working context. As well as freeform handwriting on this transparent input overlay as a sketch pad, the user can dictate some words and draw diagrams to communicate with the system.

Key words : User Interface, Input Framework, Transparent Window, Handwriting Recognition, Voice Recognition, Sketch Pad, User Test

1. Introduction

There have been endless efforts to bring in recognition-based interfaces into the existing computing machineries for decades in the name of multi-modal interface. Many commercial voice and handwriting recognition packages such as IBM ViaVoice [1] and Microsoft handwriting-recognition system [2] have shown that they are quite ready for general applications, showing highest recognition rate than ever before. Although their recognition rate is formidable, how to interface them with the applications requiring text input has not been enough explored compared to the efforts to enhance its recognition rates and speed.

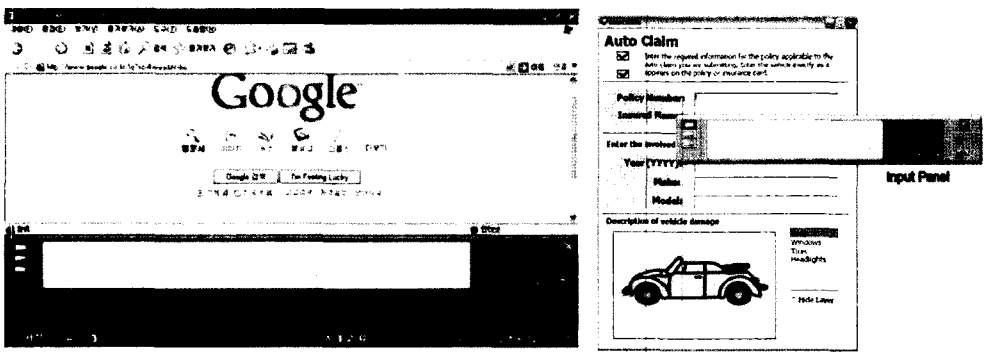
† 정 회 원 : 이화여자대학교 디지털미디어학부 연구원
cregeo@fxcode.com
†† 정 회 원 : 이화여자대학교 디지털미디어학부 교수
mjkim@ewha.ac.kr
(Corresponding author임)
††† 정 회 원 : (주) 알티캐스트 멀티프로젝트팀 상무이사
zinolee@alticast.com
논문접수 : 2007년 9월 10일
심사완료 : 2008년 3월 7일

정보과학회논문지: 소프트웨어 및 응용 제35권 제4호(2008.4)

One of the main complaints about using the pen input panel from Microsoft Tablet PC is the spatial limit for inputting as shown in Fig. 1(a). A paper, e.g., A4 size paper, lend you much more degrees of freedom for writing than the whole screen space of the pen-input display, because in general the screen resolution is far lower than the physical paper. Considering this limit of inherent screen resolution, limiting the input panel area into a system defined small window can constrain the individual writing skills of the users. Worsening the situation, the input panel is positioned far away from the actual working context such that the focus of attention of the user should repeatedly switch from one place to the other. Overall, this input scheme cannot be used for smaller devices such as PDA and mobile phone, which allows very limited area for input.

Newer handwriting input panel for the Microsoft Tablet PC can automatically place itself close to the current working window (i.e. input cursor): See a web application [2] with pen input panel but with contradictively reduced input space, as shown in Fig. 1(b). This, however, still can distract user from the very working context, a portion of which is hidden under the input panel disturbing the user from grasping the whole task. Anyway the input panel cannot be located close enough to the working window to keep the user's focus of attention in one spot. This example is just one sample that contains relatively less information so that it does not require much user insights into the working context. Moreover, it further constrains the user from freehand writing by forcing an explicit structure to the user: the pen input panel forces explicit input style all the time. Writing should be always horizontally aligned and all the recognized words in the panel should be input to the application all at once or none. It simply disables implicit structure for the user to form an idea, which is possible in practicing sketch on a paper; here the term "implicit" is so in the sense that it is perceived by the user but not the system, because it is not defined or declared to the system.

As [3] noted, even if accuracy of handwriting recognition is invariant the user acceptance of the recognition results can vary application by application. Unacceptable cases occur mostly when the user cannot properly cope with the recognition errors within the application program where the inputting error is considered critical. Therefore, the recognition errors should not immediately alter the current status of the application, which mostly likely leads to an unacceptable interface. One way to have handwriting recognition interface avoiding the situation is to have another layer only for pen input (i.e., handwriting input panel) in which user can confirm whether the recognition should go over to the application. In addition, we may bring in medi-



(a) (b)

Fig. 1 (a) Pen input panel fixed at the bottom of the desktop window on a UMPC. Note that working area of the application in focus becomes so small because the input panel takes up almost half of the screen. (b) Somewhat improved version of the pen input panel: however, it still diverts the visual attention from the working context hiding a significant portion of the working context

ation techniques [4] into the application itself or the input panel. Nevertheless, the fact every error should be corrected as in the Microsoft input panel makes using it cumbersome.

Meanwhile, researchers in [5] suggest not to use online recognition of handwriting; instead letting the users focus on taking notes instead of correcting handwriting recognition errors. In their systems notes are simply saved as they are in "ink" form containing information implicitly defined within its image and shared by the users. In [6] handwriting has been used as a manipulatable graphical object in a freeform editing system but not further processed in recognition system. Recognition task is simply delayed to the point when the user really wants it or not done at all. Note, however, that they are stand-alone applications but not interfaces to other applications, which the pen-input interface is required to be. In this paper we describe an experimental GUI attempting to address the aforementioned issues of handwriting input interface. We designed a handwriting input GUI with the following general design goals:

- Maximize the amount of screen used for inputting
- Avoid forcing the user to divert their visual attention from context
- Increase the degrees of manipulation and comfort of input

These goals were set in [7] for different styles of interactive applications too; in other words current handwriting user interfaces have not addressed commonly accepted user interface criteria so far. Our novel interface uses transparent paper as a main interface metaphor, which has been widely used in architectural design for long time as well as in modern computer graphics editing tools [7-11]. Similar to the other prior research works using the transparent medium, it acts as a buffer zone to the application, in which user can instantly write without any constraint but not losing the working context. The recognition results change the status of the application only through additional manipulation gestures in the transparent medium. As many other transparency-based applications, we separate information and interface complexity of the recognition from the application itself. Furthermore,

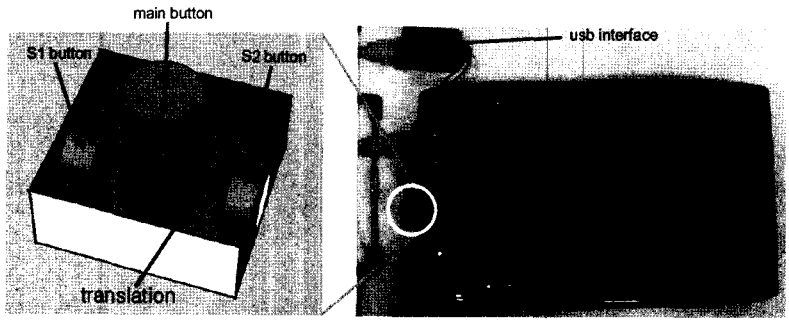
Fig. 2 A transparent window overlaid on the current desktop applications to get pen and voice input

we show that the transparent medium itself can be used as a handy note-taking interface better than usual note-taking interface.

In the remainder of this paper, Section 2 describes our handwriting interface as well as note-taking interface, while Section 3 describes how to extend the transparent input overlay to other medium interfaces: voice and diagram editing interfaces. It is followed by an evaluation of our handwriting interface methods and comparison with existing interfaces in Section 4. Section 5 further analyzes our results and summarizes them as the contributions of the research. Section 6 concludes this paper and presents some future works to be done.

2. Handwriting Interface

For our experimental setup we have used Samsung Q1 ultra mobile PC. To enable two handed interface, we physically attached a small button specially designed for the mobile PC, by which on one hand the user can invoke our new handwriting input panel as well as finalize it by pressing and releasing it, respectively. This hardware attachment consists of three buttons internally to facilitate four distinctive states as shown in Fig. 4. This style of button has not been found in existing hand-held devices; moreover, because we were to test our two handed interface making such gadgetry was inevitable. The USB device driver reads in the status of the three small buttons (button-up and -down of each button) and do the transactions according to Fig. 4. Each button(main button, s1,



(a) (b)

Fig. 3 (a) A conceptual model of a specially devised hardware button consisting of three buttons to be attached to a mobile device. (b) Attaching the button to a mobile PC

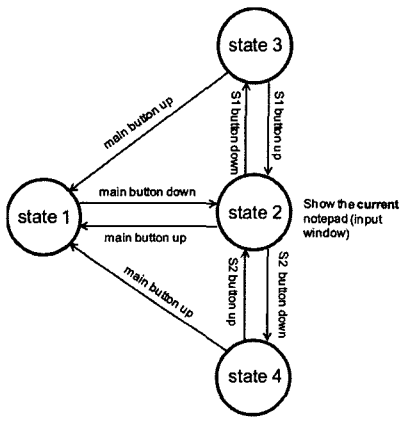


Fig. 4 Transactions about the three buttons

and s2) functions as independent button with down and up states; the main button, when it is translated back and forth, is used to push s1 and s2 button.

When the user presses the main button (state 2) a transparent window appears, as in Fig. 2, covering the whole of the display area, on top of all the other windows in the desktop. This input window stays there the same unless the user releases the button. This constraints is related to the idea of "cognitive chunk" of Bill Buxton [12]; input task should be done in a most attentive mental status of the user, i.e., while pressing something or touching. This helps the user form a cognitive chunk, avoiding many behavioral errors. When the user finally releases the button, the input window disappears. One session of handwriting is defined by this time period from pressing the button to releasing the

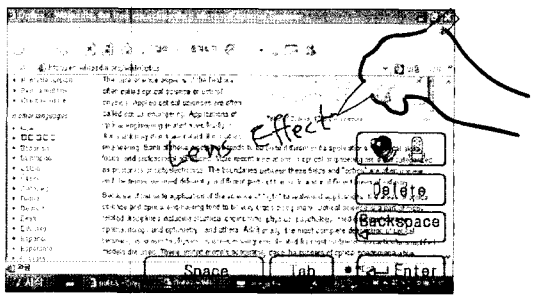


Fig. 5 Taking a note while reading a web document

button. State 3 and State 4 are left over for later use in Section 2.2, so called note-taking interface.

During the input session, user can write freely without any constraints. In the handwriting recognition system, pen traces input continuously without halting are treated as one phrase; halting for longer than a few seconds (e.g. 2 seconds) invokes recognition action from the system. Recognition results are presented right under the ink strokes. The phrase recognized forms a manipulation unit, we call this "manipulatable graphical object".

Fig. 5 demonstrates how the user can make use of the noting functionality of our interface: While reading a web document the user takes a note striking his mind. Then the ink recognized by the system and the recognition results are presented right under the ink input with a manipulation handle (See Fig. 6). It is notable that the user can write in a slanted manner, unlike the Microsoft pen input panel that requires the input to be horizontally aligned. This involves only a simple computation but provides further freedom in input style:

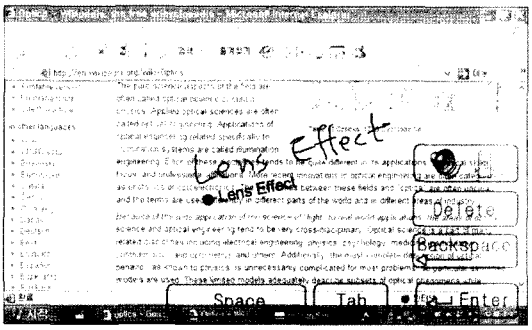


Fig. 6 Recognition results are shown as a manipulatable graphical object. A handle in front of the result is used for the user to manipulate the text later

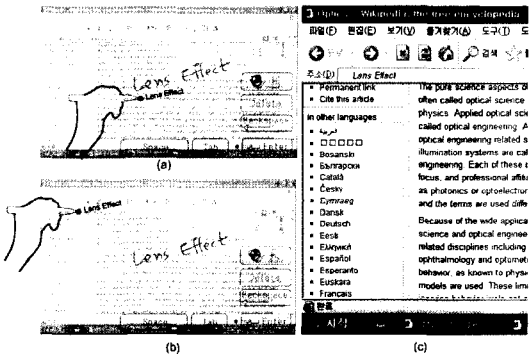


Fig. 7 (a) The handle of the text object is used to move the recognized text into the working window. (b) Dragging the handle and dropping the text object into the address bar of the internet navigator is equivalent to inputting the text. (c) Shows the navigation window after releasing the main button.

We compute the principal axis of the ink and rotates it to be aligned to the horizontal axis of the screen.

Now the user can drag and drop the results into the working window of the user's interest. The user picks the text handle and drags into the working window: In the example of Fig. 7, the user drags the text "Lens Effect" into the address window of the internet navigator. We simply copy this text into the clipboard of the window system and paste it to the address bar.

2.1 Hint-Based Mediation Interface

Our interface also provides hint-based mediation technique for error correction as shown in Fig. 8, which was introduced for the first time in [1] and further studied in [4]. In general, the goal of perfect recognition is difficult to achieve because correct recognition is best defined as what the user intents. Then, mediation techniques serve to resolve this ambiguity. User can pick any word among the recognized words to get n-best list of potential recognitions as shown in Fig. 8: In our implementation only four best potential recognitions are presented, which we thought does not complicate the interface.

2.2 Note-Taking Interface

As preluded in Section 1, the pen-input interface can be used for instant note taking too, which pen-input display users use most frequently. Our bimanual interface using the three buttons attached to the pen-input display makes note-taking task more handy. For this purpose, our interface keeps track of the notes taken in a linked list in asso-

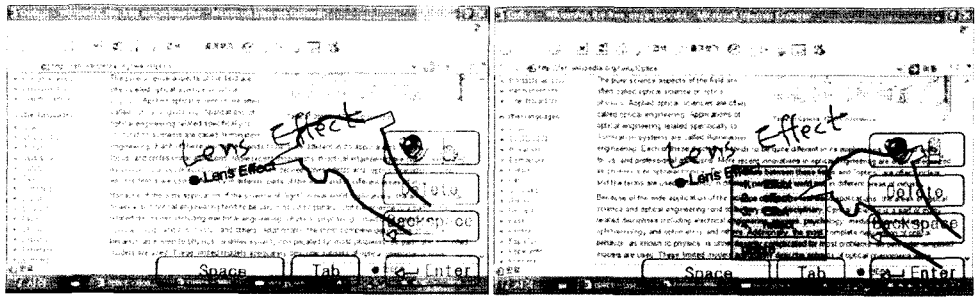


Fig. 8 (a) User picks a word, "Effect". (b) Pop-up pull down menu shows the best recognitions the recognition engine created for the handwriting

ciation with the file system.

In Fig. 4, pushing the main button down creates a new note altering the state from state 1 to state 2. Changing from state 3 to state 2 (i.e. clicking the S1 button), shows the note saved previously and save the current note into a new file. Changing the system status from state 4 to state 2 (i.e. clicking the S2 button) presents the next note in the linked list by loading the previously saved file.

Using the three buttons, users can switch from one note to the others swiftly. Furthermore, as an architect draws further details on a transparent paper laid over the other drawings, thus managing the complexity of the overall drawing, user can take the notes on top of a meaningful document laid under the transparent note. This process results in a novel and efficient note-taking interface.

The recognized texts appearing right after each recognition process, which is invoked by halting writing, may divert the user's focus of attention from the note-taking task itself. Therefore, we added a new functionality by which transparency of the recognized text can be adjusted to the user's preference: For example, see Fig. 9.

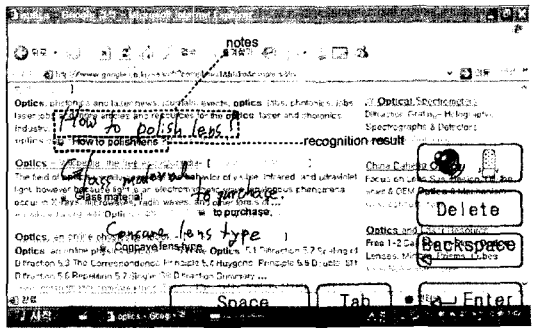


Fig. 9 Note-taking with a different transparency mode

3. Other Medium Interfaces

3.1 Voice Interface

As a multi-modal interface, our implementation supports also voice recognition interface. Since we use transparent input overlay as an intermediate layer interfacing the other applications, voice recognition, which is very sensitive to noise (i.e., machine noises, environmental sounds, voices from other persons, and so on), can be done in a selective way. While the user touches the sound icon, recognition process is invoked to accept user's voice as an input (Fig. 10). This again follows Buxton's

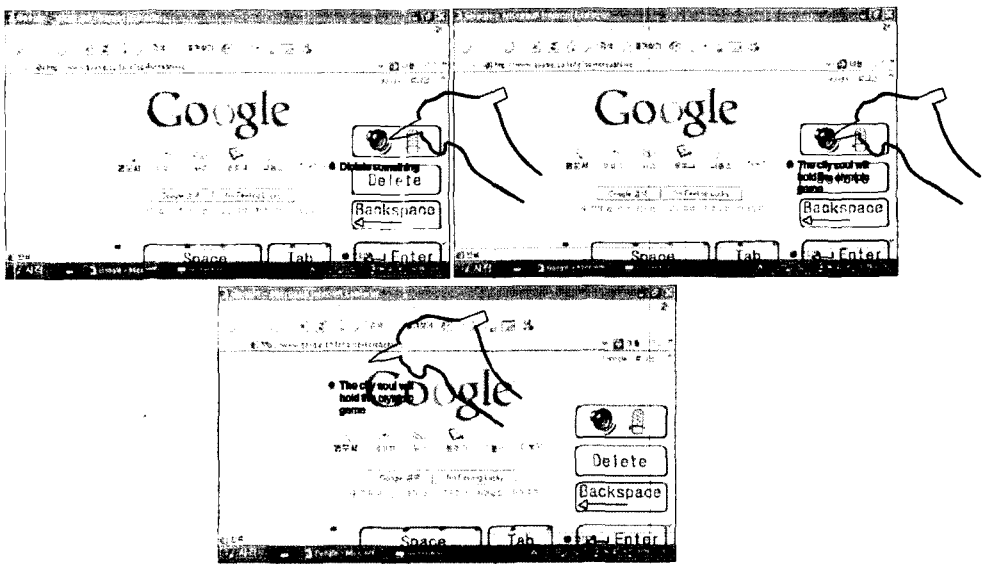


Fig. 10 Top Left: When the user touches the voice recognition icon with the stylus, the instruction, "Dictate something", appears. Top Right: The user speaks to the microphone a sentence and the recognized result is shown in response. Bottom: The user moves the stylus out of the icon region and the recognized text follows it

principle of "cognitive chunk" for recognition based applications [12].

While the user dictates to the microphone of the mobile PC during this session, recognized words are displayed right under the stylus dynamically adding new words recognized. This is mainly because interactive application should always return a certain feedback to the user so the user becomes aware of "the program is doing something according to my input behavior". When finally the stylus drags out of the area of the sound icon, the recognized text (i.e., the manipulatable graphical object) follows the stylus until the user detaches the stylus from the screen surface. Interaction with the text object afterwards is the same as what follows the text object handling after handwriting recognition.

From our various experiments we found that the voice recognition rate of the Microsoft SDK is under our expectation for a proper interactive application. Therefore we have used IBM ViaVoice SDK for our interface instead, which outperforms the former and has been widely known for its applications in many areas.

3.2 Diagram Editing

To our interest, we have used our transparent input overlay for diagram editing using gesture recognition facilities introduced in [8]. For simple implementation, we limited supported gestures within rectangle, arrow mark, circle, polygon. Since we do not have the clipboard for the diagrams, by which ASCII texts can be transferred from one application to the other, we made a simple diagram editing program that can communicate with the transparent input overlay for the generated diagrams by gesture.

While the transparent input overlay stays over the diagram editing tool, user draws the diagrams in his mind like UML diagrams: See Fig. 11. The transparent input overlay interprets the diagrams as best as it can. When the user touches the outline of the diagram, its interpretation (e.g., clean rectangle that most fits the input ink) is presented, allowing the user drags it into the position that the user wants within the editing tool. The rest of the

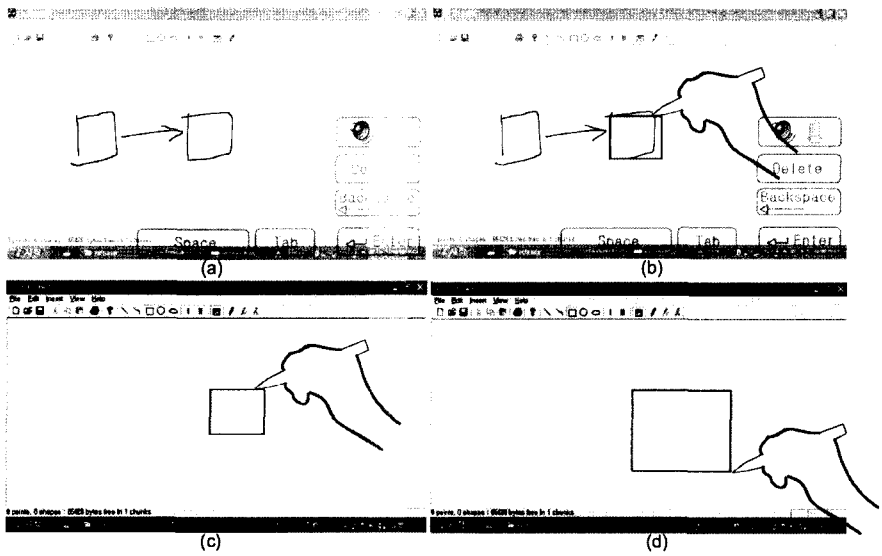


Fig. 11 (a) The user draws diagrams consisting of two rectangles and one arrow. They are properly recognized as what the user drawing intends. (b) When the stylus touches the ink of the right rectangle, the interpreted rectangle appears as clean shape. (c) Continuous dragging removes the transparent input overlay and positions the rectangle in the diagram editing program. (d) The user picks the screen surface to determine the exact position of the right bottom corner of the rectangle

shape parameters of the rectangle is edited by further picking by the user.

4. Interface Evaluation

This section reports the results of user evaluation that was conducted as a formative investigation of the effectiveness of our interface, transparent input overlay. In this evaluation, we compare our method with that of Microsoft Input Panel, for handwriting input. Comparison of the voice interface and diagram editing interface was not possible because so far we couldn't find any tool that has similar features with ours.

4.1 Subjects and Experimental Task

The participants of the test were 16 persons of mean age 25.3 years. These participants were teamed up in pairs for further discussion within the team and making a report. They were given one Samsung Q1 mobile PC, which does have only stylus as a input means, and a paper printed one long sentence with 40 words to input with the PC. They were instructed to input this sentence (1) with our interface, (2) with Microsoft Pen Input Panel as in Fig. 1(a) and (3) with Microsoft Pen Input Panel as in Fig. 1(b), ten times respectively, and to record the input time.

The reason behind the ten times of handwriting input of the same sentence is to consider learning time of the tools itself, which may vary user by user, and time to get used to the recognition style of the recognition engine used. Averaged input time then reflects the overall input performance of each group.

4.2 Quantitative Evaluation

As shown in Fig. 12, performance for the inputting task was better with our interface although the scores for the team 3, team 7, and team 8 were only slightly higher than Microsoft Pen Input Panel. According to the discussion with the subjects, this was because of their inputting style; they were writing only a few words and then confirmed to be input to the word processor they chose to use.

4.3 Qualitative Evaluation

Each group had a discussion about what are the advantages and disadvantages of using each input tools. Seven groups reported that using our input tool was more convenient to use because (1) it allows larger input region and (2) it causes less errors than the other two. Our reasoning for this phenomenon is as follows: Although we used the same recognition engine as the other two tools, our method did not restrict the user's handwriting to be aligned with the horizontal axis of the screen.

All were complaining that the first Microsoft input panel took too much space of the screen. Meanwhile, as expected, five groups were complaining that writing panel for the second Microsoft input panel was too small and restrictive for editing such a long sentence. Two groups were complaining about the response time of our interface when the user pushes the attached hardware button to invoke the transparent input overlay into the display; it takes about 0.5 seconds to respond.

5. Analysis

This section summarizes the main advantages of our interface.

• Space for input is guaranteed as large as total

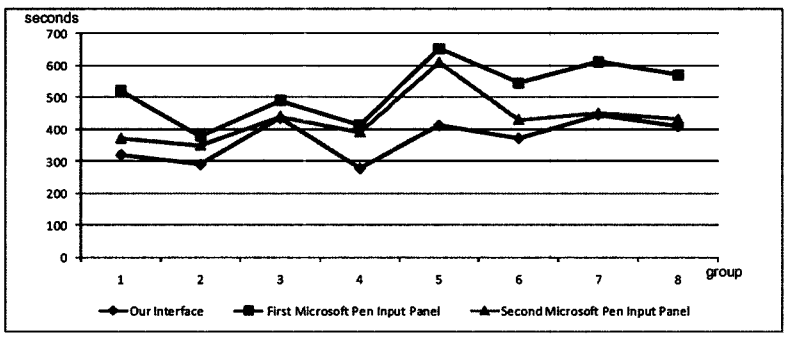


Fig. 12 User performance for inputting 40 words

resolution of the pen-input display used; this issue was problematic particularly when it comes to small mobile devices such as UMPC, PDA, and mobile phone. To simply put it, Microsoft style pen input panel cannot be easily adopted in mobile phone.

- The visual context of inputting moment can be best preserved since whole information present before the transparent input window appears is kept under the transparent input window.
- In this environment of the wider handwriting area, handwriting style of the user can be better preserved; for example, handwriting can run on a slanted way but not horizontally aligned with the display.
- User learnability issue can be improved in a certain degree; Since many phrases can be put on the input window with their respective recognition results, if the user is not satisfied with a specific writing pattern she can change the way of her writing.
- Sketchy interface becomes possible by its metaphor; Input window is regarded as an instance of transparent paper, which can be saved and recalled later. This even enhances document reading environment enabling taking note while, for example, reading complicated web documents.

6. Conclusion And Future Work

The contributions of this paper have been to present a novel pen-input interface for freehand writing and to combine this idea with voice interface seamlessly. By satisfying the design criteria of other graphical user interfaces, we could overcome, although in part, the drawbacks of the existing handwriting interface and the recognition-based applications. Moreover, our interface can be used solely for note-taking, better than existing tools, such as word processor.

Although the idea of Sec.3.2 was partly borrowed from [8], we showed that this transparent input overlay can be used in many other areas too. Being concerned that this paper becomes too long, we did not report that our approach has been used in a TV set-top box to input texts for searching online video contents with freehand writing and voice. As near future work, we will extend our implementation in Sec.3.2 for editing presentation slides; we did not yet fully utilize the advantages of the transparent input overlay in graphical object editing.

References

[1] Baumgarten. IBMrViaVoice QuickTutorial. South-Western Educational Publishing.

[2] Jeff Van West. Using Tablet PC: Handwriting Recognition. Microsoft, 2007.

[3] Clive Frankish, Richard Hull, and Pam Morgan. Recognition accuracy and user acceptance of pen interfaces. In CHI 1995 Proceedings, pages 12-18, 1995.

[4] Jennifer Mankoff, Scott E. Hudson, and Gregory D. Abowd. Interaction techniques for ambiguity resolution in recognition-based interfaces. In Proceedings of the 13th annual ACM symposium or User Interface software and technology, pages 11-20. ACM, November 2000.

[5] Richard C. Davis, James A. Landay, Victor Chen, Jonathan Huang, Rebecca B. Lee, Francis C. Li, James Lin, Charles B. Morrey III, Ben Schleimer, Morgan N. Price, and Bill N. Schilit. Notepals: lightweight note sharing by the group, for the group. In Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit, pages 338-345, 1999.

[6] Thomas P. Moran, Patric Chiu, William van Melle, and Gordon Kurtenbach. Implicit structures for pen-based systems within a freeform interaction paradigm. In Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems, pages 487-494, 1995.

[7] Gordon Kurtenbach, George Fitzmaurice, Thomas Baudel, and Bill Buxton. The design of a gui paradigm based on tablets, two-hands, and transparency. In Proceedings of the 1997 ACM Conference on Human Factors in Computing Systems, pages 35-42, 1997.

[8] Dae Hyun Kim and Myoung-Jun Kim. A new modeling interface for the peninput displays. Journal of Computer-Aided Design, 38(3):210-223, 2006.

[9] L.M. Encarnacao, O. Bimber, D. Schmalstieg, and S.D. Chandler. A translucent sketchpad for the virtual table exploring motion-based gesture recognition. Graphics Forum, 18(3):C277-C285, 1999.

[10] Jitsuro Mase. Moderato: 3d sketch cad with quick positioned working plane and texture modeling. In T.-M. Rhyne A. Chalmers, editor, eCAADe2000, Bauhaus-Universitat Weimar, pages 300-307, June 2000.

[11] M. Trinder. The computer's role in sketch design:

A transparent sketching medium. In Computers ar.d Building, Proc CAADfutures99, pages 227-244, A:lanta, 1999.
[12] William Buxton. Chunking and phrasing and the design of human-computer dialogues. Information processing, pages 1285-1291, 1986.

김 대 현

1994년 국립서울산업대학교 전산과 학사 1995년 고려대학교 컴퓨터학과 대학원 석사. 2004년 독일 Bremen 대학교 공학 박사. 1991년~1999년 한국전자통신연구원 연구원. 2004년~2006년 (재)그래픽스 연구원 선임연구원. 2006년~현재 이화여자대학교 연구원. 관심분야는 컴퓨터그래픽스

김 명 준

1989년 한국과학기술대학 전산학과 학사 1991년 한국과학기술원 전산학과 석사 1996년 한국과학기술원 전산학과 박사 1996년~1997년 University of Washington 연구원. 1997년~1998년 시스템 공학연구소 선임연구원. 1998년~2000년 한국전자통신연구원 선임연구원. 2000년~2001년 (주)엔룩스 연구소장. 2001년~현재 이화여자대학교 디지털미디어학부 부교수. 관심분야는 컴퓨터그래픽스, 물리기반 애니메이션, 게임 프로그래밍

이 진 호

1991년 2월 한국과학기술원 과학기술대학 전자계산학과 학사. 1993년 2월 포항공과대학교 전자계산학과 일반대학원 석사. 1993년 1월~2000년 5월 포항공과대학교 정보통신연구소 전임연구원. 1995년 3월~2003년 2월 포항공과대학교 전자계산학과 일반대학원 박사과정 수료. 1996년 4월~1996년 10월 서울신문사 서울신문 과학정보부 정보컬럼니스트. 1996년 7월~1996년 12월 대학문화신문사 과학정보분야 객원 전문기자. 2000년 5월~2001년 7월 주식회사 포디홈네트 설립 및 기술이사 역임. 2001년 8월~2007년 1월 주식회사 알티캐스트 경영기획실 기획담당 이사. 2007년 1월~현재 주식회사 알티캐스트 δ-project TFT 담당 상무