SpeeG: A Multimodal Speech- and Gesture-based Text Input Solution

Lode Hoste, Bruno Dumas and Beat Signer
Web & Information Systems Engineering Lab
Vrije Universiteit Brussel
Pleinlaan 2, 1050 Brussels, Belgium
{lhoste,bdumas,bsigner}@vub.ac.be

ABSTRACT

We present SpeeG, a multimodal speech- and body gesture-based text input system targeting media centres, set-top boxes and game consoles. Our controller-free zoomable user interface combines speech input with a gesture-based real-time correction of the recognised voice input. While the open source CMU Sphinx voice recogniser transforms speech input into written text, Microsoft's Kinect sensor is used for the hand gesture tracking. A modified version of the zoomable Dasher interface combines the input from Sphinx and the Kinect sensor. In contrast to existing speech error correction solutions with a clear distinction between a detection and correction phase, our innovative SpeeG text input system enables continuous real-time error correction. An evaluation of the SpeeG prototype has revealed that low error rates for a text input speed of about six words per minute can be achieved after a minimal learning phase. Moreover, in a user study SpeeG has been perceived as the fastest of all evaluated user interfaces and therefore represents a promising candidate for future controller-free text input.

Keywords

Multimodal text input, speech recognition, gesture input, kinect sensor, speeg

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Input devices and strategies*

General Terms

Human Factors, Experimentation

1. INTRODUCTION

Over the last few years, set-top boxes and game consoles have emerged as "digital life" management centres. For example, Microsoft's Xbox 360[1] console offers media centre functionality. It provides access to movie and television streaming services such as NetFlix[2], enables text and video chat and supports the interaction with Facebook, Twitter and other social networking services. Some of the tasks offered by these modern media centres, including text-based chatting, tweeting or the writing of Facebook messages, demand for efficient text input. Unfortunately, text input for set-top boxes or game consoles is often implemented as a letter by letter selection from a virtual keyboard via a game or remote controller. More recently, controller-free capturing tools such as Microsoft's Kinect[3] camera-based sensor device offer alternatives for selecting characters from a virtual on-screen keyboard. However, the overall text input process remains time consuming and error-prone which often leads to user frustration when using existing text input interfaces.

Currently, the only "solution" by manufacturers of media centres, set-top boxes and game consoles to efficiently enter text is to offer dedicated keyboards to be used for text input. For example, Microsoft offers the Xbox Chatpad[4] for their Xbox 360 and also Logitech's Mini Controller[5] can be used for inputting text on set-top boxes. Nevertheless, the state-of-the-art game consoles support a range of input modalities which can potentially be used for enhancing text input speed as well as to improve the overall user experience. Speech recognition seems to be an obvious solution for text input and an increasing number of consoles are equipped with one or multiple microphones to record a user's voice. Existing speech-based input is often limited to a small number of simple commands. An extension of the vocabulary to support more natural user interaction leads to an increase in speech recognition errors which finally might result in user frustration. The correction of erroneous recognition results has been an active research topic in the speech recognition community, and multimodal systems seem to represent a promising approach [6].

In this paper we present SpeeG, a new multimodal speech- and gesture-based text input solution targeting set-top boxes and game consoles. We start by providing an overview of existing text input interfaces for media centres. We then introduce different solutions for text input that are based on voice recognition and introduce the zoomable Dasher user interface. After introducing our SpeeG prototype which is based on Microsoft's Kinect sensor, the NITE skeleton tracking system[6], the CMU Sphinx speech recogniser [11] and the Dasher [12] user interface, we describe an exploratory user study in which SpeeG has been compared to other text input solutions. After a comparison of SpeeG with the Xbox controller-based

[1] http://www.xbox.com

[2] http://www.netflix.com
[3] http://www.xbox.com/kinect/
[4] http://tinyurl.com/xbox360chatpad/
[5] http://www.logitech.com/en-us/smarttv/accessories/devices/7539
[6] http://www.openni.org

AVI '12, May 21-25, 2012, Capri Island, Italy

virtual keyboard, the Xbox hand gesture-based text input interface as well as a speech only interface, we provide some concluding remarks and discuss potential future directions.

2. BACKGROUND

Most existing game consoles and set-top boxes offer text input via a controller or a remote control. With these controller-based interfaces, the user has to select letter by letter from a virtual keyboard as illustrated in Figure 1 for Microsoft's Xbox 360. Virtual keyboard interfaces have three main characteristics: First, they require a directional one- or two-dimensional input device to move a cursor between the various characters. Second, a confirmation button is used to select the highlighted character in order to type it in the entry box. Optionally, additional buttons can be used to delete the preceding character or to navigate between them. The third characteristic is that almost no training is required for users to understand and use these types of interfaces.

Note that the controller-based selection of characters from a virtual keyboard currently represents the de facto standard for text input on game consoles or media centres. However, text input speed via game controllers is limited to a few words per minute [15] and therefore users do not tend to write longer texts, including emails or Facebook messages, on game consoles or media centres without additional accessories such as dedicated keyboards.

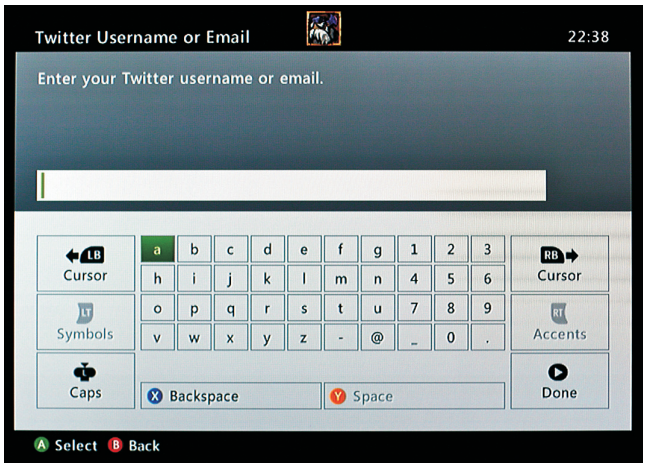

Figure 1: Xbox 360 virtual keyboard

In the same way as virtual keyboards are used for text input, speech-based text input provides a mean to replace physical keyboards. Existing speech solutions, for example the CMU Sphinx speech recogniser [11], Dragon Naturally Speaking from Nuance[7] and Microsoft's Windows Speech Recognition tool[8], transform a user's voice input into a textual representation. In order to recognise speech, users are required to complete a training phase where they are asked to read out loud a number of predefined sentences. This training is quite cumbersome for new users, especially if a user needs to perform this on multiple devices (e.g. PC, Xbox or set-top boxes). However, this training is currently absolutely necessary when dealing with non-native speakers or users with strong accents. An additional difficulty with voice recognisers is that the voice-based editing of recognition results after recognition errors requires dedicated correction commands, such as *"correct that"*, *"undo that"*, *"delete line"* or *"press enter"*.

[7]http://nuance.com/dragon/
[8]http://tinyurl.com/windows-speech

Dasher [12] is an alternative virtual keyboard technique where users have to zoom and navigate through nested graphical boxes containing single characters as illustrated in Figure 2. The static cursor represented by the cross in the centre of the screen is used to select single characters. While a user navigates in two-dimensional space, the character boxes move from the right to the left. Whenever a character box moves past the centred cross, the corresponding character is outputted to the application that has been registered for processing the text input. Figure 2 shows a screenshot of the Dasher interface for a user currently selecting the word "*Hello*".

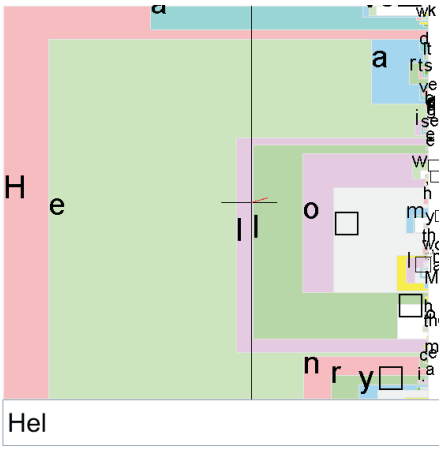

Figure 2: Dasher interface

The navigation through the character boxes is typically done via mouse input. By moving the mouse upwards or downwards, the users zooms towards the intended character boxes. When moving the mouse to the right or to the left, the navigation speed is increased or decreased. As shown in Figure 2, characters are ordered alphabetically along the vertical axis. In addition, Dasher adapts the size of character boxes depending on the probability that a given character will be the next one based on a language-specific vocabulary. Dasher has been demonstrated to be an efficient text input interface, yet needing a relatively important learning phase.

Dasher has also inspired other research projects aiming for alternative text input. In particular, projects have explored gaze direction [13, 8] and brain-computer interfaces [14] as well as two-dimensional input modes including mouse, touch screen or joystick. All these projects integrated Dasher as a direct text input tool. In contrast, Speech Dasher [10] uses Dasher as an error correction component during speech recognition. Speech Dasher first asks a user to pronounce a sentence and then, in a second step, enables them to correct the recognised sentence via Dasher.

Instead of only using a single modality for text input, multimodal user interfaces lead to more robust or more efficient text input. A potential strength of multimodal interfaces is their ability to correct recognition errors by either taking advantage of *mutual disambiguation* or by offering real-time error handling through different modalities [6]. In the context of speech recognition error correction, Suhm et al. [7] provide an overview of different multimodal error correction techniques. Novel interactive devices have since been put to use, such as touch screens [3], stylus on mobile devices [9] as well as exploratory work with virtual reality [5]. However, all these systems ask the user to review and correct speech input in a post processing step whereas our multimodal SpeeG interface enables the real-time correction of speech input.

3. SPEEG PROTOTYPE

Our multimodal SpeeG interface combines speech recognition with a continuous gesture-based real-time correction. In a typical SpeeG session, a user speaks some words which are recognised by a speech recogniser and subsequently delegated to our customised JDasher[9] interface as shown in Figure 3. In parallel, the user navigates through the words in the JDasher interface by means of hand gestures. As speech recognisers are relatively good at finding the most probable words, it is possible to offer a small list of alternative words for selection to the user. The goal of SpeeG is to support efficient text input for controller-free set-top boxes and media centres based on error-prone speech recognition and imprecise selection methods as offered by camera-based hand tracking.

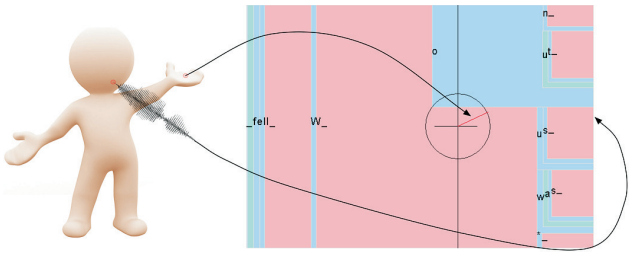

Figure 3: Speech- and gesture-based Dasher input

Our SpeeG prototype [1] mainly consists of three components forming part of a modular architecture: a speech recogniser, a hand tracking module and a modified version of the JDasher interface. All three components are fully decoupled using a unified network-based protocol for information exchange. The modularisation allows for a simple replacement of any of the three components for technical improvements such as an improved hand tracking solution or alternative speech recognisers. The modular architecture also supports the reconfiguration of our SpeeG prototype for different scenarios. For example, the visualisation might be adapted to deal with different speeds of input, screen sizes or colours. An overview of the overall data flow between the different components of the SpeeG interface is given in Figure 4.

Since the exchange of information is driven by user interaction, we will explain both the interaction steps 1 to 5 and the flow of data in the architecture shown in Figure 4 via a simple scenario. Let us assume that a user wants to enter the following text: *"My watch fell in the water"*. While the user speaks the first few words *"My watch fell"*, their words are captured by a microphone and immediately transformed into text by the speech recognition engine (1). We currently use the open source CMU Sphinx speech recognition toolkit and we have configured it for continuous speech recognition. In contrast to many other speech recognisers, Sphinx supports the output of intermediate results while the user is speaking. However, recognition rates of intermediate speech recognition results are considerably lower due to the fact that there is less chance for grammatical reasoning at the sentence level. We had to modify the Sphinx engine in order that it returns a list of alternative sound-alike words instead of a single best guess. This additional information about sound-alike words is important for our setting since we argue that:

1. A single best guess word might be incorrect since, for example, phoneme equivalent words require some sentence-level context in the decision process.

[9]http://kenai.com/projects/jdasher/

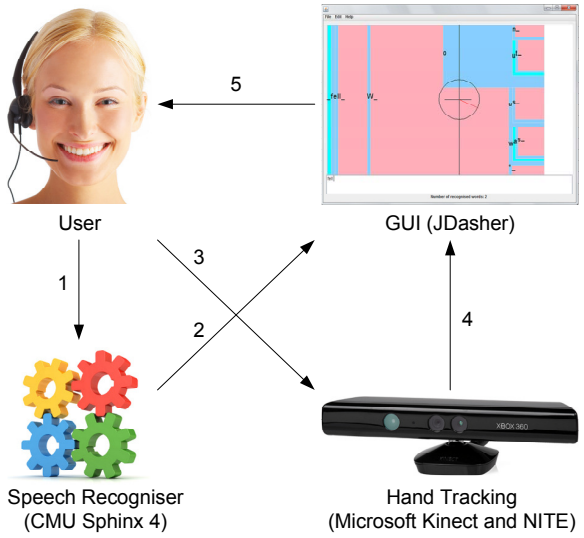

Figure 4: SpeeG data flow

2. Non-native English speaking users and users with strong accents demand for substantial additional training to improve recognition rates.

3. The recovery from potential speech recognition errors is difficult, time consuming and requires that users repeat particular words.

It should be stressed that recovering from errors in current unimodal speech recognition interfaces is time consuming and leads to user frustration. In SpeeG, we try to overcome this problem by embedding the correction phase as part of the input process. Alternative words provided by the speech recogniser are offered to the user who selects their intended words via hand gesture navigation. Our interface does not require any individual training of the speech recogniser, we support non-native English speaking users and error correction is performed in a preemptive and continuous manner. We argue that our SpeeG prototype significantly lowers the boundary for users to use speech recognition tools. The results of the speech input modality are sent to the customised JDasher interface (2). JDasher, a Java port of the Dasher interface, waits for additional user interaction captured by a Microsoft Kinect camera.

The implementation of text input via camera-based hand tracking is a challenging research topic due to the imprecise pose tracking (compared to directional pads), missing selection commands (there are no confirmation buttons) and a wide range of novice as well as expert users that should be supported by the interface. In our SpeeG prototype, the hand tracking has been implemented based on Microsoft's Kinect hardware and the NITE skeleton tracking software. NITE's software package for camera-based tracking follows the body of the user (3) and allows three-dimensional left or right hand coordinates to be translated into cursor movements. We further elaborate on this characteristic in the user evaluation when comparing SpeeG with the state-of-the-art Xbox 360 Kinect text entry method. The three-dimensional coordinates are sent to our modified version of the Dasher interface (4) which will handle the selection and visualisation process.

As mentioned earlier, Dasher is a graphical user interface where users zoom towards graphical boxes that contain a character. A character is selected by letting the character box pass by the cursor

positioned at the centre of the screen. Dasher offers a number of interesting properties for our concrete scenario:

- *Zooming*: The selection process in Dasher is simple as users *zoom* to the correct word(s) at their own pace. Each user is in control of the zooming speed as well as the direction of selection via the horizontal and vertical movements of the mouse.
- *Alphabetical order*: The Xbox virtual keyboard layout uses an alphabetically arranged keyboard that is language independent. In the same way, Dasher offers the characters to form words in alphabetical order: The letter *"a"* is found at the top, while the letter "z" is located at the bottom of the screen. This allows the user to make some general assumptions in which direction they have to navigate to go for the start of the next word before the actual words are visible on the screen.
- The *character-level* selection significantly reduces the number of possible items to select from compared to a solution where we have to select from all potential words. Words sharing the same characters are compounded and the first common characters are offered as a single choice. For example, the two words *"world"* and *"wound"* both start with the letters *"w"* and *"o"* which implies that users only have distinguish between these words at the third character. This improves readability and the selection process because it allows for bigger fonts and less cluttered character boxes.
- *Probabilities*: The size of the character boxes increases while zooming towards them. Additionally, their initial size is different based on the probability of the preceding and consecutive characters for a given vocabulary.

In addition to these graphical interface properties which were relevant for the implementation of the SpeeG interface, more graphical properties of the Dasher interface can be found in [12, 10]. In our modified Dasher interface, users navigate through the character boxes by using their left or right hand. Navigation halts when the hand is vertically aligned with the left or right hip representing the midpoint for navigating. Point gestures are used to navigate in the Dasher interface which means that moving the hand up will navigate towards the boxes at the top containing the first letters of the alphabet. On the other hand, a downward movement of the hand will navigate downwards towards the last letters of the alphabet, a white space character as well as a skip symbol (5). The skip symbol should be used when a word has been returned by the speech recogniser which should not be part of the text entry. The navigation speed is measured by how much the user moves the hand away from the centre of the horizontal axis. This means that novice users can zoom slowly through the boxes and first discover how to navigate using their hands, while expert users can navigate at high speed without having to configure the interface. In our scenario, the user can navigate to the box containing the letter *"m"* followed by the letter *"y"* to start entering the intended sentence *"My watch fell in the water"*. Depending on the accuracy of the voice recogniser, a number of alternative characters will be shown as well, for instance the letters *"h"* and *"i"* from the sound-alike word *"hi"*. For every uttered word, the user has the ability to navigate between the most probable words and typically choose between 2 to 20 alternatives depending of the results sent by the speech recogniser. Initial tests have shown that, due to the size of the speech recognition engine vocabulary as well as the diversity of accents of our users, the recognition rates dropped frequently. This forced us to offer a more extensive list of alternative sound-alike words which seamlessly maps to the "scalable" and "character-level" Dasher interface. By merging words that start with the same letters, the amount of choices the user has to make is reduced.

The original JDasher solution uses a static probability dictionary (provided at startup time) to define the width and height of character boxes. However, in our scenario we need to recalculate these probabilities on the fly based on information from the speech recogniser. This required a major refactoring of the JDasher implementation.

Finally, SpeeG is intended to provide an always-on controller-free interface. This implies that both the activation and deactivation are controlled through the application context (i.e. when requiring additional text input) or by the user (i.e. pausing or resuming the typing). A simple push gesture with the hand allows the user to explicitly signal the beginning of new input, which helps to avoid any unintended behaviour.

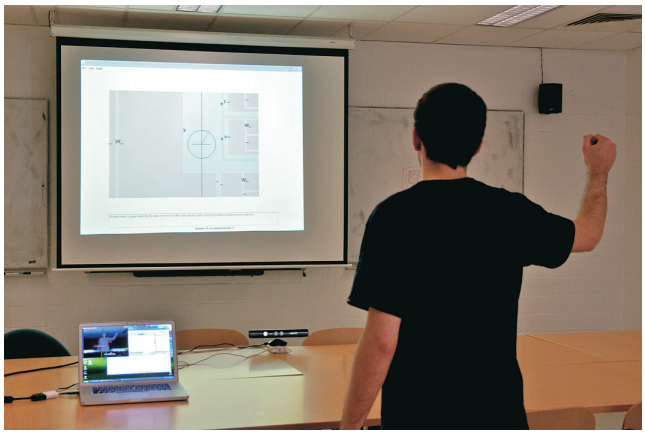

Figure 5: SpeeG prototype setup

To summarise, SpeeG is a controller-free zoomable user interface that enables the real-time correction of a voice recogniser via gesture-based navigation through the most probable speech recognition results. Our SpeeG prototype, which is shown in use in Figure 5, deals with error correction in a preemptive and continuous manner. This real-time error correction represents a major difference from state-of-the art speech-based text input solutions, which perform a second correction step. By resizing the character boxes based on voice input, we guide the user to select their intended uttered words. As multiple words per utterance are displayed, we overcome the major problem with speech recognition tools failing to deliver perfectly accurate full vocabulary text input for controller-free interfaces. Due to the fact that the character boxes get quite big closely before being selected, SpeeG minimises unintended mis-selections, even when using an imprecise control mechanism. By offering a zoomable user interface, the user is also in control of the navigation and interaction speed, which helps to support both novice and expert users without additional configuration parameters.

4. EXPLORATORY USER STUDY

In order to assess the user interest in our approach compared to existing text input solutions for consoles and set-top-boxes, an exploratory user study has been conducted in our laboratory. The user study was based on the following scenario: the user is in front of their television set and needs to enter a short sequence of words

(about 30 characters) separated by blank spaces. Typical real-world examples of this scenario would include the typing of a query for the Bing search engine on the Xbox 360 console or the writing of a Twitter message on a television set-top box such as Google TV[10]. In such use cases, text is often entered through a virtual on-screen keyboard shown on a television and characters are selected via the remote control or a console gamepad. Speech recognition software represents an alternative to virtual keyboards. However, high error rates and the low tolerance to changing ambient noise conditions often lead to a frustrating user experience. Our assumptions are that our SpeeG solution might provide a less frustrating alternative to existing solutions, while at least achieving a comparable performance. The goal of the study presented in this section was to verify these two hypotheses. As SpeeG is still a laboratory prototype, the study took the form of an exploratory user study in order to assess the weak and strong points of our approach compared to other text input methods currently used in game consoles and set-top boxes.

In particular, SpeeG was compared to three alternative text input approaches:

- Text input via controller-based character-by-character selection on a virtual keyboard as discussed earlier in the background section when introducing the Xbox 360 virtual keyboard (see Figure 1). This method represents the de facto standard text input method for game consoles and set-top boxes. For our study, we used the virtual keyboard interface offered by the Xbox 360 game console in combination with the game console controller.

- Gesture-based character-by-character selection. This is a relatively new interaction mode for text input in some commercial products. The Kinect-based interface to perform Bing searches, which forms part of the latest version of the Microsoft Xbox 360 operating system[11], was used for this study.

- Speech recognition software. The Sphinx speech recognition software has been used for the speech-only interface in our exploratory user study, since it is the speech recognition software that has also been used in SpeeG.

4.1 Methodology

Participating subjects were asked to type a set of six sentences in these four different setups: 1) using an Xbox 360 controller and the virtual keyboard of the Xbox 360 "Dashboard" interface, 2) using the Kinect-based gesture interface, 3) using the Sphinx speech recogniser and 4) using our SpeeG text input interface. The order in which the participants used the different text input interfaces was uniformly distributed. However, the same six sentences *S1* to *S6* were used in the same order for all setups:

- S1: "*This was easy for us.*"
- S2: "*He will allow a rare lie.*"
- S3: "*Did you eat yet.*"
- S4: "*My watch fell in the water.*"
- S5: "*The world is a stage.*"
- S6: "*Peek out the window.*"

[10]http://www.google.com/tv/
[11]Xbox Firmware 2.0.14699.0

Out of these six sentences, the last three originate from the set proposed by MacKenzie and Soukoreff [4] while the first three sentences were selected from DARPA's TIMIT [2], which is a common resource in the speech recognition community. Both sets were used in order to have representative sentences from the text input community and the speech recognition community. To compute the word per minute (WPM) rate, a "word" was considered as a 5-character sequence.

The Xbox 360 Dashboard virtual keyboard takes the form of an on-screen alpha keyboard, as shown earlier in Figure 1. Letters are arranged on a 7x4 grid and the currently selectable letter is highlighted in green. Users navigate through the letters by using the controller's d-pad and select a letter using the controller's 'A' button. The selected letter is subsequently displayed in the white text field in the upper part of the window. The 'X' button can be used to correct a wrongly entered letter.

The Kinect-based gesture interface is also part of the operating system of the Xbox 360 console. As shown in Figure 6, the screen is divided into different zones. Letters are arranged as a one-dimensional array in the upper part. To input text, a user has to make a "push" gesture on each letter with a hand icon towards the white text field in the upper part of the screen. The hand icon is controlled by the player's hand position and detected via the Kinect infrared camera. When approaching this hand icon to the array of letters, the array will be zoomed on a set of 10 letters to simplify the selection of a letter. Users typically need to move the hand up for about 20-40 cm to bring the chosen letter to the text field. Note that, similar to the virtual keyboard interface, this text input interface forms part of the user interface of a commercially widely available product.

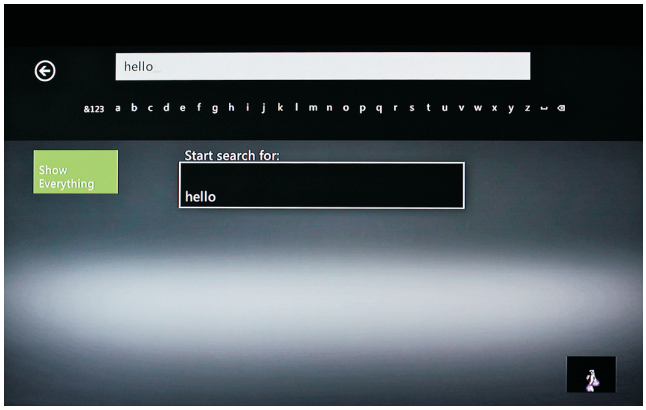

Figure 6: Xbox 360 gesture-based text input interface

The Sphinx speech recogniser setup has been tested by asking users to read sentences aloud word per word. On the screen, the users could see which word was recognised by Sphinx and they were able to correct a wrongly recognised word by repeating the word. Sphinx was parameterised to recognise an open vocabulary represented by the Darpa Wall Street Journal (WSJ) Trigram LM containing twenty thousand words. If the speech recogniser did not recognise the correct word after five attempts, the user was asked to move on. For the speech only interface we used a standard omnidirectional microphone which was positioned one meter in front of the user.

In the SpeeG setup, users were presented the SpeeG application and used the interface as it has been presented in the previous section. Note that the Sphinx and SpeeG setups used exactly the same parameters for the speech recognition engine.

Before inputting the sentences *S1* to *S6* with each of the four setups, the study participants were allowed to train with each of the interfaces until they felt comfortable using it. This training phase lasted between one and four minutes, depending on the user. Each setup was tested with the user seated or standing in front of the image projected on a screen via a beamer, based on the fact whether the interface required the user to stand in front the system.

After the evaluation with the four setups, participants were given a questionnaire asking them to rate the "quality" of each of the four setups on a six-level Likert scale and we asked them which interface they found

- the easiest to learn,
- the easiest to use,
- the fastest to enter text,
- which interface was their preferred one.

Finally, a text field allowed the participants to enter some general comments.

4.2 Study Participants

Seven male participants aged 23 to 31 were asked to participate in the laboratory user study. The participants were computer science students or researchers, with two of them being HCI practitioners. Two users had expert knowledge with the Xbox 360 controller while two other users had passing knowledge. Three users indicated passing knowledge with speech recognition software, while the remaining users had no experience at all. No user had any prior experience with the Dasher interface. All subjects had vision corrected to normal and were right handed. Even if all users were fluent in English, there were no native English speakers. Two users were native French speakers (*user 3* and *user 5*), whereas the remaining five participants were native Dutch speakers. For a typical speech recogniser, coping with non-native accent is a challenging task. Therefore, this user group was very interesting to test the real-time speech error correction offered by SpeeG. However, this influenced only two out of the four setups since the controller-only and Kinect-only input do not make use of any speech.

4.3 Results and Discussion

The mean input speed results of our user study are highlighted via the graphs shown in Figure 7.

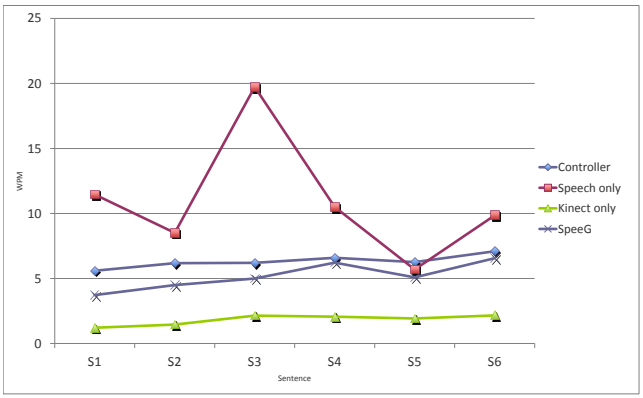

Figure 7: Mean WPM per sentence and input device

The results for the Xbox 360 virtual keyboard setup with an average input speed of 6.32 words per minute (WPM) are slightly higher than the average 5.79 WPM observed by Wilson et al. [15] for a similar setup in 2006. Note that four of our users had either passing or expert knowledge with the Xbox 360 virtual keyboard interface, which might explain the difference, as the users in the study by Wilson et al. had no prior experience.

As shown in Figure 8 with the detailed results for the Xbox 360 virtual keyboard setup, five out of seven users finished the evaluation with a similar performance and non-expert users were getting close to the performance of expert users.

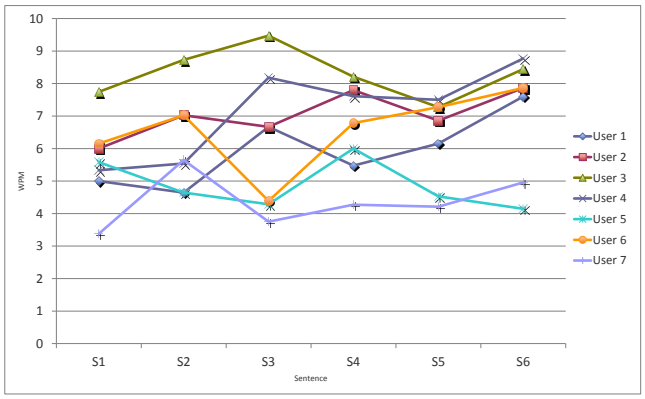

Figure 8: WPM per user and sentence for the controller and virtual keyboard setup

Figure 9 shows the results for the Kinect-only setup where there seems to be less discrepancy between the different users. However, one has to take into account that two out of five users (*user 1* and *user 6*) refused to finish the evaluation for the Kinect-only setup after one or two sentences only, because the interface was too frustrating. For the five users who finished the task, the total mean writing speed of 1.83 WPM as well as the high error rate illustrated in Figure 11 are a testament to the difficulties encountered by the study participants. Almost all users reported heavy weariness in their arms, as well as major frustration with the setup of the interface. In particular, users were puzzled that only one hand was used, that speech had no influence at all and that no word prediction or any other kind of help was available. This very low WPM performance is even more surprising when taking into account that the Kinect-only setup represents the latest interface for the commercially widely available Xbox 360.

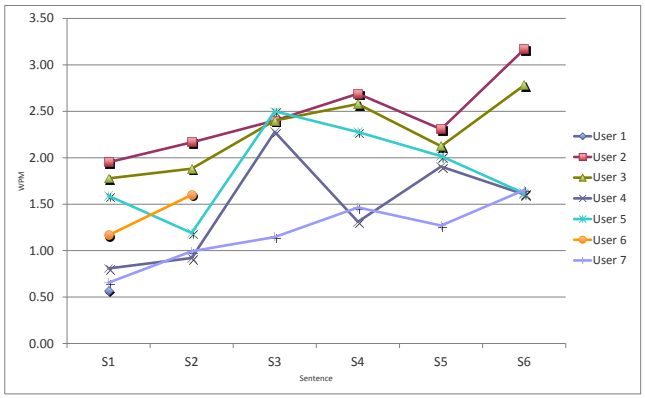

Figure 9: WPM per user and sentence for the Kinect-only setup

In the speech-only tests, we achieved strongly varying results with an overall mean WPM of 10.96. The major differences between single sentences and users are illustrated in Figure 10. The different difficulties of the sentences is well reflected in the final error rate per sentence which is highlighted in Figure 11. For example, sentence *S3* with its short and well-defined words posed no major difficulties to the study participants. In contrast, sentences *S2* or *S5* lead to more wrongly recognised words. In particular, words like *"rare"* and *"world"* posed major problems to the speech recogniser, resulting in a high number of false positives as reflected in Figure 11. Also note that the two native French speakers (*user 3* and *user 5*) achieved a lower performance than the Dutch speakers. We believe that this might be caused by differences in pronunciation between people used to a Latin language and the ones using a West Germanic language which is pronunciation-wise closer to English.

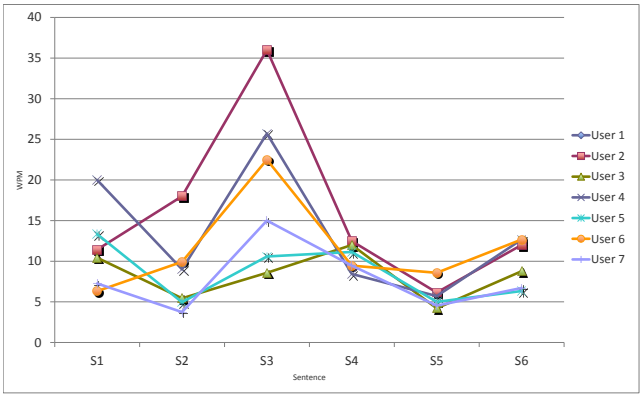

Figure 10: WPM per user and sentence for the speech-only setup

In comparison to the speech-only setup, the SpeeG error rate starts lower for sentence *S1* and decreases to zero for all users from the third sentence onward as illustrated in Figure 11. The average text input speed for the six sentences with SpeeG is 5.18 WPM as highlighted in Figure 12. Most users showed a steady improvement of the text input speed throughout the six sentence, with one user increasing from 2.67 to 7.86 WPM between the second and the sixth sentence, reflecting a three times improvement within a few minutes only. This improvement throughout the sessions reflects the learning curve of SpeeG. First, the learning curve originates from the Dasher interface, as already observed by Ward et al. [12]. Another relevant factor influencing the learning curve is the multimodal aspect of SpeeG, which increases the level of concentration needed to efficiently use the interface. Nevertheless, a majority of users quickly achieved text input rates between 7 and 9 WPM. In parallel to this user study, an expert user reached a mean text input speed of 11.5 WPM on the same set of sentences by using the SpeeG interface. As a comparison, the best expert user for the Xbox virtual keyboard setup managed to reach a mean of 8.38 WPM on the six sentences. There seems to be a great potential for improvement based on our SpeeG interface.

A very interesting observation lies in the discrepancy between the participants' opinion about the fastest interface and the concrete measurements. Five out of seven users experienced SpeeG as the fastest interface, while the two remaining users thought that the Xbox 360 virtual keyboard was the best performing method. None of the participants voted for the speech-only interface as the

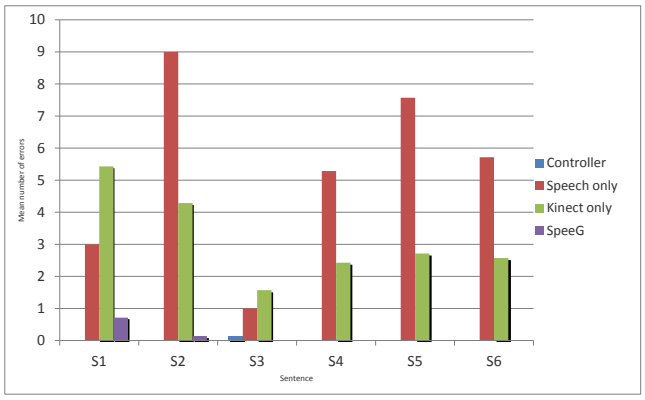

Figure 11: Mean number of errors per sentence

fasted input method, while in reality it was the best performing interface. We assume that the user frustration experienced due to speech recognition errors had a negative impact on the participants' assessment of the subjective efficiency. On the other hand, our SpeeG prototype gave users the feeling that they were in control. A total of four users also specifically reported the "fun" and game-like aspect of SpeeG while navigating through partial recognition results. This is also reflected by the fact that five out of seven users preferred SpeeG over the three other user interfaces.

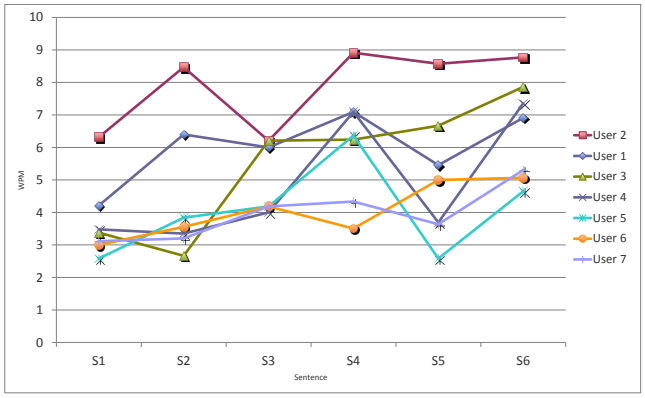

Figure 12: WPM per user and sentence for the SpeeG setup

To conclude, the SpeeG interface is faster than the dedicated gesture-based interface of the Xbox 360 console and is close to the performance of the virtual keyboard interface of the same console, with however greater variability between subjects. Speech recognition only is the fastest text input interface, but frustration seems to play a role in a user's estimation of the best performing interface.

5. FUTURE WORK AND CONCLUSION

In the future, we plan to reduce the physical strain imposed by the current SpeeG interface since some users reported that they got tired after waving their hands for a long time. In a next version of SpeeG, we therefore plan to investigate smaller gestures, such as using mainly the fingers, to control the interface. It might also be interesting to evaluate SpeeG with the original Dasher interface controlled via the same gestures but without any speech recognition. Such an evaluation could also be used to explore the cognitive load depending of the involved modalities and the task at hand.

Adding real-time supervised learning capabilities to SpeeG could also be interesting since all speech input is already annotated and corrected by the users themselves. This opens the possibility to greatly improve the speech recognition results by creating a user-specific acoustic model while the user is interacting. This would allow for reducing the number of alternative words. An interesting path to explore is how we could optimise our interface to reduce the required hand movement in this case. For instance, the interface could give more importance to a highly probable sequence of characters.

The missing support to enter names via SpeeG is a limitation which could be overcome by offering a character-level mode where users can freely select the letters they require. The entering or exiting of the character-level mode can be achieved by adding a special character box at the bottom of the selection process. Another possibility to switch between modes would be to make use of symbolic gestures performed by the second hand. Currently, SpeeG only requires single-hand interaction which allows us to integrate such an option in the future. Symbolic gestures could invoke special functionality such as "*confirm word*", "*skip word*", "*replace word*" or "*enter character-mode*", thus adding an "expert mode" to speed up the text entry.

In this paper, we have presented SpeeG, a multimodal interface combining speech recognition with a gesture-based real-time error correction for text input. SpeeG uses a modified version of the Dasher interface and presents the user a set of potential words detected by the speech recogniser. A user can navigate through the different word possibilities to select the right word or just skip a word and move on to the next one. In an initial laboratory user study of our innovative SpeeG prototype with seven users, we assessed an average text input rate of 6.52 WPM over a number of sessions. However, in comparison to the high number of errors for the speech-only input, the error rates almost dropped to zero in the SpeeG setup.

Our user study has also revealed a strong potential for improvement due to the learning curve of SpeeG, with one user ultimately reaching up to 11.5 WPM. An interesting fact of our study was that users experienced SpeeG as the most efficient interface even if in reality two other solutions performed better in terms of the entered words per minute. This very positive feedback might also result from the fact that users had the feeling of being in control of the interface as well as the game-like character of SpeeG reported by several participants. The reduction of user frustration seems to play a key role in a user's perception of the user interface efficiency.

The promising results after a minimal learning phase in combination with the overall user satisfaction confirms that our multimodal SpeeG solution has a strong potential for future controller-free text input for set-top boxes, game consoles as well as media centres.

6. ACKNOWLEDGMENTS

We would like to thank all the study participants. Furthermore, we thank Jorn De Baerdemaeker for implementing major parts of the SpeeG prototype and Keith Vertanen for his helpful comments. The work of Lode Hoste is funded by an IWT doctoral scholarship. Bruno Dumas is supported by MobiCraNT, a project forming part of the Strategic Platforms programme by the Brussels Institute for Research and Innovation (Innoviris).

7. REFERENCES

[1] J. D. Baerdemaeker. SpeeG: A Speech- and Gesture-based Text Input Device. Master's thesis, Web & Information Systems Engineering Lab, Vrije Universiteit Brussel, September 2011.

[2] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, N. L. Dahlgren, and V. Zue. TIMIT Acoustic-Phonetic Continuous Speech Corpus. Linguistic Data Consortium, Philadelphia, 1993.

[3] D. Huggins-Daines and A. I. Rudnicky. Interactive ASR Error Correction for Touchscreen Devices. In *Demo Proceedings of ACL 2008, Annual Meeting of the Association for Computational Linguistics*, pages 17–19, Columbus, USA, 2008.

[4] S. I. MacKenzie and W. R. Soukoreff. Phrase Sets for Evaluating Text Entry Techniques. In *Extended Abstracts of CHI 2003, ACM Conference on Human Factors in Computing Systems*, pages 754–755, Fort Lauderdale, USA, April 2003.

[5] N. Osawa and Y. Y. Sugimoto. Multimodal Text Input in an Immersive Environment. In *Proceedings of ICAT 2002, 12th International Conference on Artificial Reality and Telexistence*, pages 85–92, Tokyo, Japan, December 2002.

[6] S. Oviatt. Taming Recognition Errors with a Multimodal Interface. *Communications of the ACM*, 43:45–51, September 2000.

[7] B. Suhm, B. Myers, and A. Waibel. Multimodal Error Correction for Speech User Interfaces. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 8:60–98, March 2001.

[8] O. Tuisku, P. Majaranta, P. Isokoski, and K.-J. Räihä. Now Dasher! Dash Away!: Longitudinal Study of Fast Text Entry by Eye Gaze. In *Proceedings of ETRA 2008, International Symposium on Eye Tracking Research & Applications*, pages 19–26, Savannah, USA, March 2008.

[9] K. Vertanen and P. O. Kristensson. Parakeet: A Continuous Speech Recognition System for Mobile Touch-Screen Devices. In *Proceedings of IUI 2009, International Conference on Intelligent User Interfaces*, pages 237–246, Sanibel Island, USA, February 2009.

[10] K. Vertanen and D. J. MacKay. Speech Dasher: Fast Writing using Speech and Gaze. In *Proceedings of CHI 2010, ACM Conference on Human Factors in Computing Systems*, pages 595–598, Atlanta, USA, April 2010.

[11] W. Walker, P. Lamere, P. Kwok, B. Raj, R. Singh, E. Gouvea, P. Wolf, and J. Woelfel. Sphinx-4: A Flexible Open Source Framework for Speech Recognition. Technical Report TR-2004-139, Sun Microsystems Inc., November 2004.

[12] D. J. Ward, A. F. Blackwell, and D. J. C. MacKay. Dasher – A Data Entry Interface Using Continuous Gestures and Language Models. In *Proceedings of UIST 2000, 13th Annual ACM Symposium on User Interface Software and Technology*, pages 129–137, San Diego, USA, November 2000.

[13] D. J. Ward and D. J. C. MacKay. Fast Hands-free Writing by Gaze Direction. *Nature*, 418(6900):838, August 2002.

[14] S. Wills and D. MacKay. Dasher – An Efficient Writing System for Brain-Computer Interfaces? *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(2):244–246, June 2006.

[15] A. D. Wilson and M. Agrawala. Text Entry Using a Dual Joystick Game Controller. In *Proceedings of CHI 2006, ACM Conference on Human Factors in Computing Systems*, pages 475–478, Montréal, Canada, April 2006.