SpeeG2: A Speech- and Gesture-based Interface for Efficient Controller-free Text Entry

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

ABSTRACT

With the emergence of smart TVs, set-top boxes and public information screens over the last few years, there is an increasing demand to no longer use these appliances only for passive output. These devices can also be used to do text-based web search as well as other tasks which require some form of text input. However, the design of text entry interfaces for efficient input on such appliances represents a major challenge. With current virtual keyboard solutions we only achieve an average text input rate of 5.79 words per minute (WPM) while the average typing speed on a traditional keyboard is 38 WPM. Furthermore, so-called controller-free appliances such as Samsung's Smart TV or Microsoft's Xbox Kinect result in even lower average text input rates. We present SpeeG2, a multimodal text entry solution combining speech recognition with gesture-based error correction. Four innovative prototypes for the efficient controller-free text entry have been developed and evaluated. A quantitative evaluation of our SpeeG2 text entry solution revealed that the best of our four prototypes achieves an average input rate of 21.04 WPM (without errors), outperforming current state-of-the-art solutions for controller-free text input.

Categories and Subject Descriptors

H.5.2. [**Information Interfaces and Presentation (e.g. HCI)**]: User Interfaces

General Terms

Design, Experimentation, Human Factors

Keywords

SpeeG2; speech input; gesture interaction; multimodal input; text entry; camera-based UI

1. INTRODUCTION

Since the early days of computing, the field of text entry systems has been dominated by keyboards. This is caused by the fact that text entry solutions have the trade-off between efficiency and training time as well as between the device size and the character set size [12]. The widespread familiarity with QWERTY keyboards causes the training time to go down and efficiency to go up, which makes keyboards a very efficient text entry solution.

However, in many situations it is either not possible or not optimal to provide a full-fledged keyboard [4, 9, 7]. This is strengthened by the recent deployment of research prototypes in the real world such as gesture keyboards for mobile devices. Similar to mobile devices, smart TVs and set-top boxes allow users to browse for additional information on the Web, tweet[1] or review movies while watching a television show [6]. The keypad of remote controllers contains buttons with letters that enable a simple form of text input. Nevertheless, with the introduction of controller-free appliances including Samsung's Smart TV[2], Microsoft's Xbox Kinect[3] or recent large public displays and Internet-enabled embedded devices, new challenges arise to provide adequate text entry methods.

Current text input for set-top boxes makes use of a virtual keyboard which is normally navigated via a D-pad controller (as available for the Microsoft Xbox, Nintendo Wii or Sony Playstation 3) containing one or multiple discrete confirmation buttons. The performance of virtual keyboard input was measured to be between 5.79 and 6.32 words per minute (WPM) [13, 2]. However, these solutions require physical buttons to select and confirm characters, which is inherently problematic for controller-free text entry. First, it is hard to find a balance between accidental activation and hovering over an option for some time. Second, the performance would be worse due to the timeout period to confirm every single word.

For decades, speech has been the holy grail when it comes to fast text entry. Besides being intuitive and non-intrusive, its performance characteristics are very promising as speech input can easily reach input rates of 196 WPM [14], while the average typing speed on a keyboard is only 38 WPM [3]. On the other hand, the inaccuracy of speech recognisers requires human correction which poses major challenges for the adoption of speech-based text entry. In addition, speech-based correction frequently results in cascading errors and therefore an extra input modality is at least advised [10]. Most multimodal interfaces of this kind use a two-phase text entry approach. In a first phase, the user decides when the speech recording starts and stops by pressing a button. In the second phase, the user corrects any errors made by the speech recogniser via mouse or keyboard. Note that the switching between these two phases not only reduces the potential performance but also requires an intrusive physical controller.

ICMI'13, December 9–12, 2013, Sydney, Australia

ACM 978-1-4503-2129-7/13/12 ...$15.00.
http://dx.doi.org/10.1145/2522848.2522861.

[1] `http://www.twitter.com`
[2] `http://www.samsung.com/us/2012-smart-tv/`
[3] `http://www.xbox.com/kinect`

SpeeG2 provides a solution for controller-free text entry by fusing information from speech and gesture input. Speech is used as the main modality to enter text, while gesture input from the dominant hand is used to correct the speech recognition results. We have developed and evaluated four prototypes to overcome the lack of discrete input. Some of the key design elements of our work, such as the grid layout, simple gestures to confirm correct results and the support of continuous interaction are based on earlier experiments in various related work.

Commercial systems for controller-free text entry currently only reach input rates of 1.83 WPM [2]. With the increasing interest from industry (e.g. Microsoft's Xbox Kinect, smart TVs) and the emergence of ubiquitous controller-free intelligent environments, there is a need for new efficient ways of controller-free text entry. We start in Section 2 by presenting related work in the domain of speech- and gesture-based text entry solutions. This is followed by our general design decisions regarding the SpeeG2 user interface in Section 3. The functionality offered by our four different SpeeG2 text entry prototypes is introduced in Section 4. After presenting the results of a quantitative and qualitative evaluation of these four prototypes, we provide some final conclusions.

2. RELATED WORK

Speech Dasher supports writing text through a combination of speech and the Dasher user interface [11]. Their work extends Dasher by visualising the output of a speech recogniser rather than single letters. The speech-based output consists of n-best word candidates that can be selected, resulting in a higher text input rate. Based on a two-step model, the user first utters a sentence and then presses a discrete button to disable the microphone before switching to the correction phase where the recognised sentence can be modified via the zoomable Dasher interface [12]. The Speech Dasher prototype also targets users with limited mobility and is used on a personal computer system since the correction phase can be controlled via mouse or gaze.

The output of a speech recogniser offers many different possible candidates for one utterance. In Speech Dasher, only the top predictions are directly added to the interface. Excluded options and character-based selection can be enabled by selecting a dedicated star character ($\star$). The developers made this design choice because too many alternative choices would increase the difficulty in navigating the interface. Speech Dasher uses the Sphinx[4] speech recogniser with a trigram language model. Depending on the participant, a UK or US acoustic model is applied. Three participants were used in the user study and the average text entry rate was 40 WPM compared to the 20 WPM of the original Dasher interface. The word error rate (WER) for Dasher was 1.3% while the WER for Speech Dasher was 1.8%. Note that these numbers are optimised using user-specific native speech recognition training. The performance for a non-native English speaker (i.e. German) was not as good due to the fact that the recogniser used a US acoustic model. Furthermore, the visualisation was not optimal for viewing at a distance and the use of discrete start and end buttons does not map well controller-free input.

SpeeG (v1.0) is a system similar to Speech Dasher but with a focus on controller-free imprecise gesture-based input for devices such as always-on set-top boxes, game consoles or media centers [2]. The SpeeG prototype offers a non-intrusive multimodal user interface that combines speech and gestures in a Dasher environment. While speech is used as the primary input, pointing gestures are used for the correction phase. However, instead of the two-phase model introduced by Speech Dasher, SpeeG uses a continuous model which allows users to continue speaking while they are using gestures to correct previously recognised words. A quantitative evaluation of SpeeG demonstrated that this model was able to achieve 6.52 WPM which is comparable to the performance achieved with a virtual keyboard in combination with a game controller (between 5.79 WPM [13] and 6.32 WPM [2]). In the qualitative study of SpeeG, users suggested to offer a word-level selection to further improve the performance and to reduce the ergonomic issues such as fatigue. We argue that the physical strain not only originates from using a mid-air interface, but due to the use of the Dasher-based interface which requires users to point into a certain direction for a longer period of time.

Parakeet [10] combines speech and touch to enter text on mobile devices. It works in a two-step process: first the sentence is recorded and when the discrete 'Mic off' button is pressed, a touch interface is presented to correct the hypothesis of the speech recogniser. In contrast to zoomable Dasher-based solutions, Parakeet uses a grid layout to present the speech recogniser's hypothesis. The key design decisions were fragmented interaction, the avoidance of cascading errors and the exploitation of alternative recognition results. The user interface grid consists of columns representing consecutive words and rows that offer the n-best word candidates from the speech recogniser. The bottom row additionally presents a delete block which is used to skip invalid word candidates. A virtual keyboard can be used as a fallback solution to enter words that are not present in the speech recognition vocabulary. Note that other systems such as SpeeG do not offer such fallback functionality. The touch-based interface of Parakeet requires discrete commands to switch between the two-step model. Unfortunately, discrete commands are time consuming and non-intuitive for camera-based gesture input.

Sim [8] describes an interface for combining speech and touch gestures for continuous mobile text entry. Experimental results show that concurrent speech input enhances the accuracy of a gesture keyboard even in noisy conditions. However, their interface requires precise input to select characters using a swipe gesture. Additionally, in case of errors, the user is still required to correct the words on a character level using a virtual keyboard.

3. SPEEG2 DESIGN

SpeeG2 is a multimodal, controller-free text entry solution using speech as the main input modality and hand gestures to correct or confirm the proposed entry. Based on related work and the scenario described in the introduction, we identified several interesting challenges: (1) provide a continuous model for both the speech recognition and hand gesture-based interaction to eliminate the need for discrete buttons, (2) reduce physical strain by allowing rest positions for the arms, (3) optimise the performance in terms of WPM and WER. In this section, we discuss the general concepts, architecture and control flow of SpeeG2, while the four different prototypes to perform the word selection are described in Section 4.

3.1 Architecture and Control Flow

Our four prototypes share the same interaction with the speech recogniser and the skeletal tracking of the Kinect sensor. The difference between the prototypes lies in the user interaction when correcting speech recognition errors (selection process). The common architecture shared by all four prototypes is illustrated in Figure 1.

First, a user utters a sentence (1) and the speech recogniser translates the spoken sentence into a sequence of words (2). At any time when a user speaks, the SpeeG2 GUI visualises what the speech

[4]http://cmusphinx.sourceforge.net

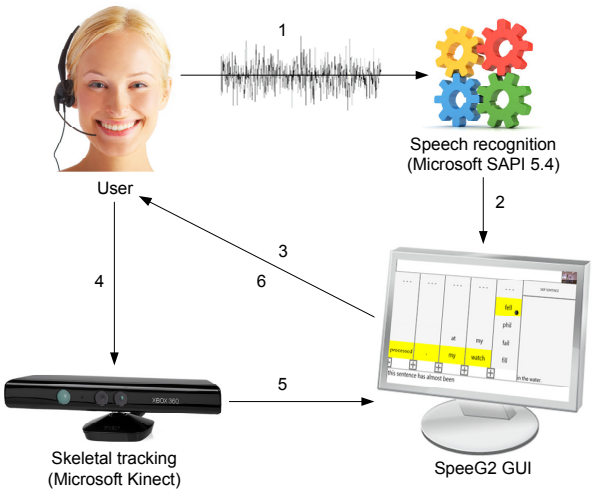

Figure 1: SpeeG2 interaction

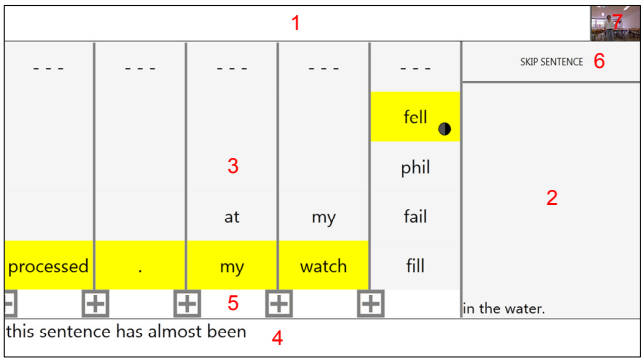

Figure 2: Highlighted parts of the grid layout

recogniser assumes to be the correct word sequence. Even if a user has not yet finished a sentence, partial results are shown in the GUI. When a sentence is spoken, the selection process becomes active (3). The user can start correcting the recognised word sequence by using the dominant hand as input modality (4). The hand movement is registered by a Microsoft Kinect and transformed to screen coordinates (5). Via the GUI the user gets continuous feedback about the speech recognition and the hand tracking (6). Note that the communication between the speech recogniser and the GUI has been realised via asynchronous network communication. This allows for abstraction and independent evolution of both components and depending on the scenario, our solution might be tailored with domain-specific speech recognition. A more detailed description of the inner workings of the GUI is provided in Section 3.2.

Due to the continuous nature of both, speech- and gesture-based input, our interface was built to support sequence (1), (2), (3) independently from the sequence (4), (5), (6). Therefore, speech input and gesture-based correction can overlap and occur in parallel, providing more freedom to the user and potentially improving the performance. A user can also first speak a few sentences forming a paragraph and perform the corrections afterwards.

3.2 Interaction Features

The graphical grid layout user interface of SpeeG2 entails a number of important features for this kind of multimodal application. In the following, we briefly describe the different components of the SpeeG2 graphical user interface which are highlighted by the numbers (1) to (7) in Figure 2. Note that all the red annotations shown in the following screenshots do not form part of the user interface.

(1) *Visualising what a user is saying:*
The top area visualises intermediate speech results. Instead of waiting for a full sentence candidate from the speech recogniser, we use this *direct feedback* to let a user know that the system is listening. Therefore, all words in this area are susceptible to changes depending on the grammar rules applied by the speech recogniser. It is common to see this direct feedback pattern in dictation software since it improves the connection between the system and the user. As soon as the speech recogniser has sent its final hypothesis, the colour of the best sentence candidate will change from black to green. In SpeeG2, a valid sentence has to consist of a sequence of at least three words. If the system detects a word sequence of less than three words, the corresponding text will be coloured red and not considered as a sentence. We decided to define such a minimal sentence length in order to filter out noise and short unintended utterances. After a valid sentence has been recognised by the speech recogniser, the text will be queued in the area indicated by number 2 in Figure 2.

(2) *What still needs to be processed:*
This GUI element forms a queue for all accepted input that still has to be corrected or confirmed. In the example shown in Figure 2 *"in the water."* forms part of the sentence *"My watch fell in the water."* which is being corrected. This allows the user to build larger paragraphs and also to use speech and gesture input in parallel. However, if a user first speaks a few sentences before starting to correct them, this part of the screen helps to remember what has already been said, allowing *delayed correction* while providing *memorisation*.

(3) *Processing speech recognition results in a grid layout:*
The grid area contains all the words that can be substituted with other word candidates via hand gestures. As speech recognition is nearly accurate but not perfect, most of the classification errors can easily be corrected by choosing one of the alternative hypotheses. The columns of the grid represent a word sequence and the rows offer *alternative word substitutes*. In the example shown in Figure 2, two sentences are being processed. The first sentence is *"This sentence has almost been processed."* and the second one is *"My watch fell in the water."*. Figure 2 shows the state where the last two elements of the first sentence (*"processed ."*)—in this case without any alternative word candidates—and the beginning of the next sentence (*"my watch fell"*) have to be corrected. In the fifth column, the speech recogniser found *"fill"* to be more likely than *"phil"*, *"fail"* and *"fell"*. The word candidates returned by the speech recogniser are sorted in inverse order meaning that the bottom row contains the most likely words. This allows the user to confirm a correctly recognised sentence with minimal effort by holding the dominant hand as low as possible without having to move it up or down. To form the correct sentence, the user has to correct *"fill"* to *"fell"* by selecting the element in the second row of that column. The top row of each column offers the possibility to skip and *delete* a particular word by selecting the *"- - -"* element. Note that a full stop is also considered a word in order to provide a clear separation between sentences. The grid highlights selected words with a yellow background. However, the way in which the user interacts with the grid and

selects alternative words is different for each of the four prototypes and will be presented later. After a sequence of words has been confirmed or corrected, it is sent to the area showing the processed parts which is highlighted by number (4). The size of the grid areas depends on the size of the display or other contextual parameters (e.g. the distance from the screen) and should be large enough to also work with less accurate hand movements. To insert missing words, users can make use of the insertion feature represented by area (5).

(4) *What has been processed:*
The area highlighted by number (4) contains all words that have been processed via the grid. It therefore shows the final corrected text and should be seen as the actual entry box to external applications. A current limitation of SpeeG2 is that text in this box cannot be altered anymore.

(5) *Insert word(s):*
The plus signs are used to insert a (single) missing word between two columns. This *insertion* feature is activated by hovering over the plus sign for 700 milliseconds and opens the insert dialogue box as shown in Figure 3. The insert feature works with the concept of speech reiteration. Only a single word that the user utters is shown in the dialogue box. If the recognised word is incorrect, the user simply needs to utter the word again. Note that this solution is susceptible to cascading errors. However, if the speech recognition engine is not able to correctly recognise a word after multiple trials, alternative correction methods (such as spelling mode) can be used. After a word is confirmed to be inserted, it is added at the location of the selected plus sign. This feature enables users to address scenarios where the speech recogniser ignored a specific word. Since this feature is not frequently used, we opted to use a discrete action (i.e. hovering by using the hand). While hovering over a plus sign button, the black cursor fills up orange to visualise the timeout, a mechanism that is commonly used for camera-based applications.

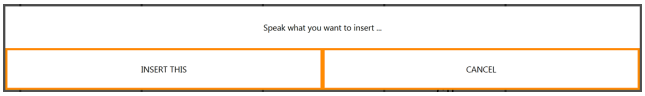

Figure 3: The insert dialogue box

(6) *Skip sentence:*
If the speech recogniser returned mostly incorrect results or even captured unintended input, a user can use the *skip sentence* feature. When area (6) is hovered over for 700 milliseconds, the current sentence will be deleted. To distinguish between multiple partial sentences, the active sentence is identified based on the number of words in the grid. In Figure 2 most words originate from the sentence *"My watch fell in the water"*, meaning that this is the active sentence.

(7) *Camera visualisation:*
When designing interactive camera-based applications, it is important to provide users feedback about their position and the tracking accuracy. Sensors such as the Microsoft Kinect have a limited field of view and depth range. Therefore, area (7) shows the camera feed in order to enable users to correct improper positioning or solve other kinds of problems such as inadequate lighting conditions.

3.3 Spelling Mode

Previous features show how SpeeG2 deals with substitution, insertion and deletion errors in the speech recognition hypothesis. However, some words like the names of people are not likely going to be part of the speech recognition vocabulary. To offer a complete text entry solution, SpeeG2 provides a spelling mode where words can be spelled out. This mode can also be used when non-native English speakers continuously mispronounce a particular word. The spelling mode works as a substitution method for an invalid word and is activated by a *push up* gesture with the non-dominant hand. The grid component is then transformed from a word-based to a character-based selection. All other GUI elements such as the insertion buttons, the skip element or feedback views remain intact. A user can now spell the word and the rows in the grid will provide candidate letters instead of words. Furthermore, the spelling mode provides a natural language feature allowing users to elaborate on their spelling by using words starting with a specific letter. For example, a user might say *"a as in alpha"* to clarify the character *"a"*. Note that the spelling mode can also be used to slightly modify a candidate word. For instance, to add an *"s"* at the end of a word, the user activates the spelling mode and then uses the existing insertion feature to add an *"s"* character. As illustrated in Figure 4, the spelling mode is visualised by purple column borders, a single letter in each column and a special *"*end*"* block at the end of a word.

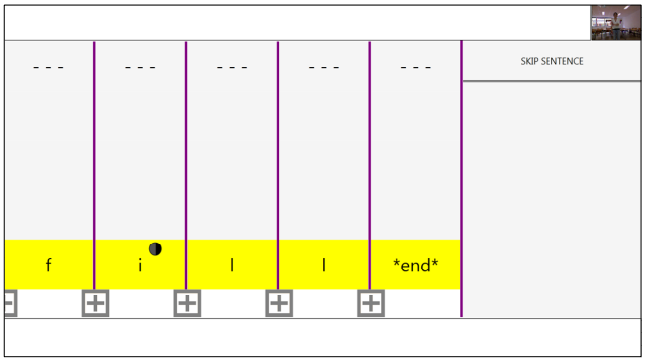

Figure 4: Correcting the word *"fill"* in spelling mode

The selection process of the grid in spelling mode works the same way as each prototype works in non-spelling mode. However in spelling mode, each time a user utters a letter, it fills the currently active column with the most probable letters.

3.4 Accuracy and Training

Previous work suggests to increase recognition accuracy by training the speech recogniser and generating a profile for each user. While speech accuracy is very important, the main goal of SpeeG2 is to provide a text entry tool which is immediately usable by a large number of people, even if training might improve the overall performance as demonstrated by Parakeet [10]. One of our main intentions is to avoid asking people to go through a training phase before they can start entering text. This barrier might also be one of the reasons why dictation software is not used very often, especially when users own a multitude of electronic devices. Therefore, SpeeG2 uses a generic model offered by Microsoft's Speech Recognition API to reach a large group of users. Nevertheless, users who are willing to invest some time in a training phase can build a speech profile by using existing tools. We argue that even without any training and for non-native speakers, current

state-of-the-art generic speech recognisers including Microsoft's and Google's speech recognition, provide adequate candidate results. Driven by results from previous work, we further opted for sentence-level recognition to improve the recognition rates due to the use of a language model.

4. SPEEG2 PROTOTYPES

We introduce four different prototypes which share a common grid interface but offer different forms of interaction for correcting speech recognition errors. We evaluated different selection strategies in the setup shown in Figure 5 and observed whether accidental triggering is an issue in some of the prototypes.

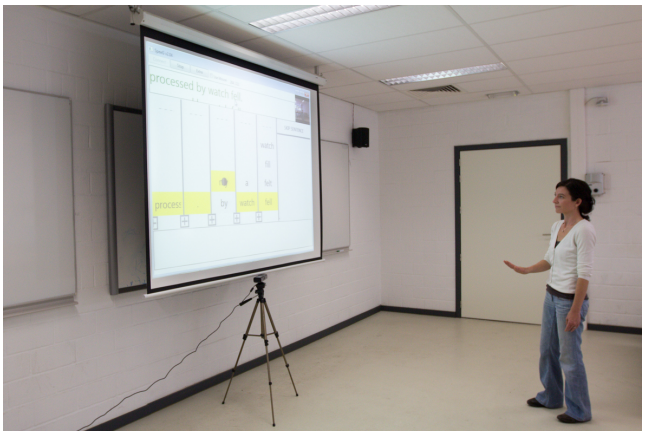

Figure 5: User interacting with one of the SpeeG2 prototypes

4.1 Scroller Prototype

The Scroller prototype uses similar design concepts as the Dasher interface. The interface shown in Figure 6, is controlled by navigating towards the next word in a stepwise manner. The scrolling steps are represented by the numbers -2 to 3 which have been added to the screenshot. When the progress bar at the top is filled (i.e. is fully green), a step occurs and the next word is put into the active column (0). The speed at which the progress bar fills is controlled by the horizontal movement of the dominant hand. The further away the hand is from the body, the faster the progress bar fills.

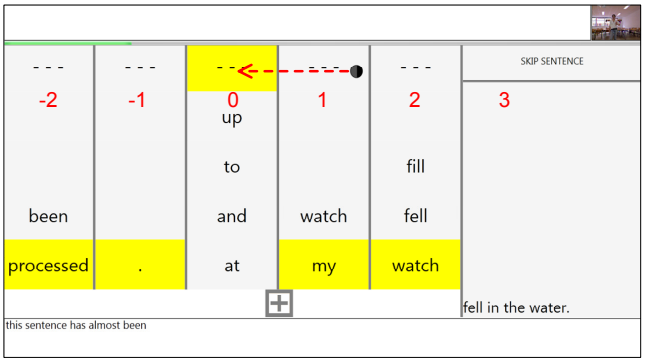

Figure 6: Scroller prototype interface

The user is also allowed to go back to previously confirmed words by moving the dominant hand to the other side of the body. For example, when the right hand is moved to the left side of the body the progress bar will reduce its value and we start to go backwards to the previous step when reaching the active column (0). A vertical movement of the hand is used to choose between the candidate words within the active column. The other columns are used to visualise consecutive and previous words.

The Scroller prototype reuses some concepts from Dasher to deal with inaccurate input in an incremental manner. However, compared to our earlier SpeeG (v1.0) prototype, it reduces physical strain as users select words instead of letters and are able to relax their hand which causes the progress bar to halt. Note that the spelling mode is available for all four prototypes and whenever a slight modification has to be performed, the spelling mode can be activated by performing a *push up* gesture with the non-dominant hand. Furthermore, to insert a word before the currently active column, the user can hover over the *plus sign* below the grid.

4.2 Scroller Auto Prototype

A variation of the Scroller prototype is the Scroller Auto prototype shown in Figure 7. The difference is that the green progress bar has been removed and the movement occurs continuously. Instead of processing the words in a step-by-step manner, the columns move sideways (similar to a 2D side-scrolling game).

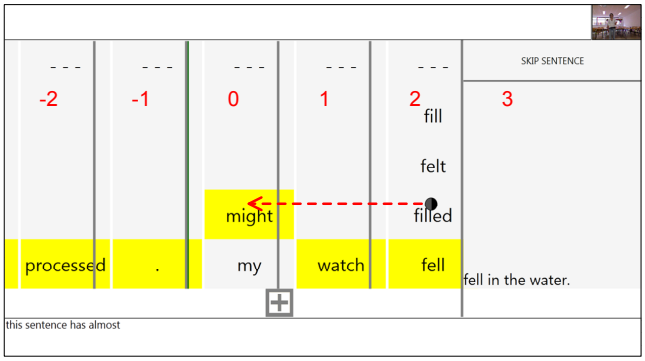

Figure 7: Scroller Auto prototype interface

Moving the dominant hand on the x-axis still controls the speed while vertical hand movements are used to select a word within the active column. The active column is the column currently in the centre (0). In the example shown in Figure 7, the column with the words *"might"* and *"my"* is active and *"might"* is selected because the cursor (represented by the black dot) is horizontally aligned with it (as indicated by the arrow). The location of the cursor (in column 2) is currently far to the right of the centre, implying a high scrolling speed. Whenever words cross the centre, they are confirmed and the next word can be corrected.

4.3 Typewriter Prototype

The Typewriter prototype is based on the concept of traditional typewriters where the carriage had to be pushed back to start writing on a new line. Even though typewriters are seldom used nowadays, people still remember them and know how they used to work. We wanted to exploit this knowledge in order to minimise the learning time and to increase usability. In the Typewriter prototype, the selection of a word is not dependent on an active column anymore. Instead, a single swipe of the hand can select an entire sequence of words on the grid. This is illustrated by the red arrow in Figure 8 selecting the words *"processed"*, *"."*, *"my"*, *"watch"* and *"fell"* in the current view. Once the red area on the right-hand side is reached, the words are committed and the following set of queued words are shown on the grid similar to a new line on a typewriter.

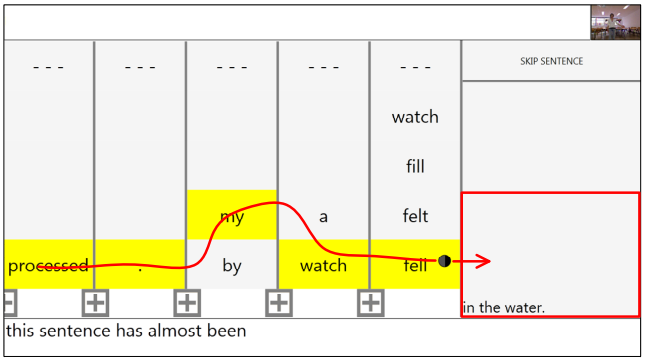

Figure 8: Navigating through the Typewriter interface

This prototype was optimised to navigate very fast through the hypothesis space. The downside of this approach is that committed words can no longer be edited. Additionally, the red area does not require any hovering time since the words are confirmed as soon as it is hit. Therefore, we were concerned about the accidental activation of a "carriage return" and introduced a slight variation with the Typewriter Drag prototype.

4.4 Typewriter Drag Prototype

The Typewriter Drag prototype extends the Typewriter prototype by requiring an explicit drag movement after the red zone shown at the right-hand side of Figure 8 is reached. Similar to a manual carriage return on old typewriter machines, the dominant hand has to be moved back to the left hand side. This means that the selection is done from left to right and the result is confirmed by *dragging* it to the left. This gesture reduces the potential risk of an accidental activation of a carriage return. Furthermore, errors can be undone by dragging in the opposite direction. Once the dragging is activated, the columns that were processed change colour to visualise the confirmation process. A drag can always be interrupted and the confirmation can be cancelled by moving the hand into the opposite direction.

5. EVALUATION STRATEGY

To evaluate SpeeG2 and the four proposed prototypes, we conducted a quantitative and qualitative pilot study. All users got an introduction and a maximum training period of five minutes for all prototypes before the evaluation was conducted. This enabled users to get used to the speech recognition engine and how their pronunciation influences the results. It also let them get comfortable with the different SpeeG2 prototypes. Note that we further uniformly distributed the order of the tested prototypes. Last but not least, the same nine participants were used to evaluate the *speech only* solution and the four SpeeG2 prototypes.

5.1 Participants and Method

The study featured nine participants (aged between 20 and 40 years). Seven participants had a computer science background but nobody used voice recognition on a frequent basis and 67% of the participants have never used speech recognition. Among the participants there were eight native Dutch speaking and one French user. The participants had a variety of different accents coming from different parts in Belgium. All tests were performed with the generic English US acoustic model of the Microsoft Speech Recogniser 8.0. As argued in the introduction, our goal is to provide a generic text entry solution that requires no training at all. Furthermore, future work could incorporate some form of automated training based on the user-corrected speech utterances. By relying on face recognition for user identification and continuous training, users could further improve their performance. However, this is out of the scope of this paper and we focus on a multimodal text entry solution supporting non-native users without any necessary configuration phase. In our setup shown earlier in Figure 5, we used a regular projector that is capable of offering the same kind of visualisation as provided by a large TV. Users were positioned 2.5 metres away from the screen as proposed by the Xbox Kinect scenario. During the development, initial tests suggested that offering four candidate words per column provided a good balance between offering enough candidates, dealing with the imprecise hand tracking of the Kinect and the visibility of the text at such a distance. In the qualitative study, no user suggested to change this configuration. Due to some limitations in terms of the availability of a facility, the study was conducted in a noisy air conditioned room forcing us to use a Sennheiser PC 21-II headset. However, in a less noise polluted environment and with a good sound recording device such as the one offered by the Microsoft Kinect, similar results should be achieved. After the introduction and a short training session, all participants were asked to learn the following six sentences by heart in order that later no reading interruptions or hesitation occurred: *"This was easy for us."* ($S1$), *"He will allow a rare lie."* ($S2$), *"Did you eat yet."* ($S3$), *"My watch fell in the water."* ($S4$), *"The world is a stage."* ($S5$) and *"Peek out the window."* ($S6$). Note that three sentences originate from DARPA's TIMIT [1] speech recognition benchmark and the others from MacKenzie and Soukoreff [1, 5] to evaluate text entry performance.

5.2 Performance Measurement

In each test, we recorded the time it took the participants to process the sentence starting from the point when the first sound of the sentence was uttered until the time when the entire sentence (including the full stop) was processed. To compute the text entry speed, the standard measure for a word was used. A word is considered a five character sequence, including spaces and punctuation. The text entry speed is computed in words per minute (WPM). In addition, the word error rate (WER) was computed. The WER measure was computed as the word-level edit distance between the stimulus text and a user's response text, divided by the number of characters in the stimulus text.

For each participant, the time they spent with *alternative correction methods* (e.g. insertion dialogue or spelling mode) was also recorded. The goal was to observe whether users needed these correction methods and whether they contribute to the reduction of the WER. When speech recognition performed poorly, the option to skip the sentence was allowed as well (without resetting the observed time).

5.3 Speech as a Single Modality

To measure the amount of errors and potential speed without the use of extra modalities, a *speech only* test was conducted. When the sentence was recognised correctly, users were allowed to continue to the next sentence. If one or multiple errors occurred in the result, the number of errors were noted down. Participants were asked to repeat the sentence up to a maximum of five times when errors occurred. The number of tries participants needed was recorded together with the average number of errors in a sentence.

5.4 Prototypes

To test the performance of the four prototypes, each participant was asked to test every single prototype in addition to the speech-only test. First, a prototype was chosen uniformly distributed over

all participants. However, in order to reduce confusion, every time the Scroller prototype was chosen it was followed by its closely related Scroller Auto prototype and vice versa. The same strategy was applied for the Typewriter and Typewriter Drag prototypes. Then participants were asked to produce the same six sentences $S1$ to $S6$ that were also used for the speech only test. Again, for each sentence, the time and the number of errors made by the speech recogniser were recorded and we also noted down which feature the participant used to correct the sentence. After this quantitative study, a qualitative study was conducted to investigate the usability of the four SpeeG2 prototypes. Each participant was asked to fill in a questionnaire about their experience with the prototypes and the speech recognition.

6. QUANTITATIVE RESULTS

6.1 Overview

The mean WPM and WER for each prototype are highlighted in Table 1. The highest mean WPM was achieved in the speech-only test. However, there was no correction phase besides repeating a sentence. Therefore, the WER of the speech-only test should be observed as the error score after correction. The WER of other tests (Scroller, Scroller Auto, Typewriter and Typewriter Drag) shows the WER before correction. After correction, all SpeeG2 prototypes resulted in a WER of 0% for all participants.

	Speech	Scroller	Scroller Auto	Type-writer	Type-writer Drag
WPM	77.63	13.59	9.69	21.04	15.31
WPM-SD	9.71	3.12	3.22	6.50	3.99
BC-WER(%)	17.72	20.43	25.68	20.15	17.47
BC-WER-SD	12.16	15.80	14.10	11.63	11.51

Table 1: Average per participant WPM and WER before correction (BC-WER) for each prototype together with the corresponding standard deviation (SD)

We verified our data using a general linear model repeated measures analysis with the different solutions (Scroller, Scroller Auto, Typewriter and Typewriter Drag) as within-subject variable. The results show a significant effect of SpeeG2 on the WPM count ($F(4, 24) = 24.91, p < 0.001$). A post-hoc analysis shows that the Scroller performance was significantly higher than the Scroller Auto ($p = 0.035$) performance, but significantly lower than Typewriter ($p = 0.002$). The Scroller Auto performance was significantly lower than Typewriter ($p = 0.001$) and Typewriter Drag ($p = 0.003$). The Typewriter performance was significantly higher than Typewriter Drag ($p = 0.010$). Furthermore, Scroller and Typewriter Drag did not differ significantly. Our quantitative data shows that the Typewriter prototype is indeed the best performing interface with a mean text entry speed of 21.04 WPM (standard deviation $SD = 6.85$). This can also be observed in the box plot shown in Figure 9, highlighting the WPM for each prototype.

The speech recognition accuracy varied from person to person and greatly depended on the user's accent and pronunciation. In particular, one participant suffered from bad accuracy with a WER of up to 100% for sentence $S2$. The mean WER before correction for all prototypes and participants was 20.29% ($SD = 12.61$). This is comparable to one error occurring in a five word sentence. Considering that all participants were non-native English speakers, this is better than the error rate of 46.7% for non-native participants in the Speech Dasher study mentioned earlier.

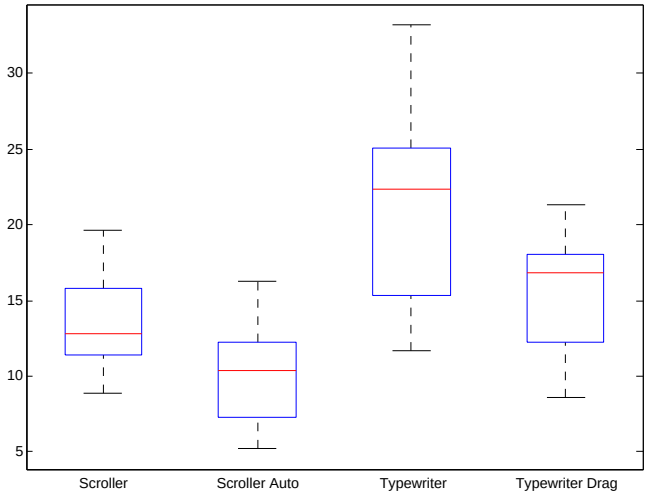

Figure 9: Box plot of WPM per prototype

6.2 Discussion

We would like to further discuss some observations that we made based on our data and state some interesting elements about the evaluation. The proposal of n-best word candidates ($n = 4$ in our setup) has shown to be a valuable asset as it was used for nearly all sentences. We also observed the frequent use of the spelling mode for slight modifications of words. The Typewriter prototype did indeed suffer from accidental confirmation, however this was only during the training phase and not during the evaluation. Further, the use of the explicit drag movement in the Typewriter Drag prototype to go back to previously confirmed words was not used, while in the Scroller prototype users did go back to a previous word a few times. The skip sentence feature was used when the speech recognition results were completely invalid. This was rather unexpected as it was designed to skip irrelevant or noisy conversations. The insertion mode was the least frequently used correction method which implies that users did not experience issues related to the use of linear correction (i.e. word by word selection). Overall, users were able to use the different means of correction methods. We also observed that they sometimes did not use the most optimal correction method (as observed from an expert point of view) which might be improved if SpeeG2 is used more frequently.

During the study, participants were asked to perform the tests sentence by sentence such that we could study the results in more detail. Therefore, users did not entirely benefit from the parallel processing capabilities potentially offered by SpeeG2. However, we argue that the parallel input allows more freedom for different scenarios: if the user is not focussed enough they can just speak an entire paragraph and correct it later, or an expert user can use this feature to exploit higher performance. We did observe participants using both speech and gesture input at the same time: e.g. while they were hovering for 700 milliseconds to activate the skip sentence button, they already started uttering the sentence such that they could start correcting as fast as possible. It should be noted that the presented numbers are obtained in a worst case scenario where non-native English speakers were asked to enter text without speech recognition training. We assume the WPM could be further improved when performing the optional speech recognition training with native as well as non-native English speakers. However, a main goal of SpeeG2 is to allow any user to start using the interface without requiring any prior configuration or training steps, thereby reducing potential boundaries of speech technology adoption.

7. QUALITATIVE RESULTS

Our qualitative questionnaire (using a Likert scale from 1 to 6 ranging from *"not agree at all"* or *"very bad"* to *"completely agree"* or *"very good"*) consisted of 29 questions investigating previous experience with speech recognition as well as the quality of certain aspects of each prototype. The users found that the speech recognition results combined with their alternative candidates were decent (3). Two participants agreed that they experienced physical strain (4), while the others declared a score of 1 or 2 (disagree) which is quite an improvement compared to our earlier results achieved with SpeeG (v1.0). The qualitative results confirm the performance of the Typewriter prototype. It was evaluated as the easiest to use and considered to be the fastest prototype to enter text. Only one participant found the Typewriter Drag faster which was in fact confirmed by their quantitative measurements. Five participants preferred using the Typewriter prototype. The remaining three preferred the Typewriter Drag prototype. As potential improvements, participants suggested to add more alternative word choices to the prototypes. However, we will have to investigate this since it might reduce the readability. Furthermore, three participants would like to see an improvement in the speech recognition.

8. CONCLUSION

We have presented SpeeG2, a multimodal speech- and gesture-based user interface for efficient controller-free text entry. A formative quantitative user study revealed that our Typewriter prototype reaches an average of 21.04 WPM, which outperforms existing solutions for speech- and camera-based text input. Furthermore, the Typewriter prototype was also the preferred prototype of our participants as demonstrated by a qualitative evaluation. The highest recorded speed for entering a sentence was with the Typewriter prototype at a rate of 46.29 WPM, while existing controller-free text entry solutions such as SpeeG (6.52 WPM) and the Xbox Kinect Keyboard (1.83 WPM) are far less efficient. Interestingly enough, our controller-free multimodal text entry solution also outperforms game controller-based solutions (5.79–6.32 WPM) [13, 2]. Furthermore, the grid-based user interface layout of SpeeG2 reduces physical strain by not requiring continuous pointing as required in Dasher-like solutions. Last but not least, all participants were able to produce error free text entry via the effective multimodal combination of speech and gestures offered by SpeeG2. We hope that our promising results will lead to further research on multimodal speech- and gesture-based interfaces for emerging ubiquitous environments, smart TVs and other appliances with a demand for controller-free text entry.

9. ACKNOWLEDGMENTS

We would like to thank all the study participants. Furthermore, we thank Sven De Kock for implementing major parts of the presented SpeeG2 prototypes. The work of Lode Hoste is funded by an IWT doctoral scholarship.

10. REFERENCES

[1] J. Garofolo, L. Lamel, W. Fisher, J. Fiscus, D. Pallett, and N. Dahlgren. TIMIT Acoustic Phonetic Continuous Speech Corpus, 1993.

[2] L. Hoste, B. Dumas, and B. Signer. SpeeG: A Multimodal Speech- and Gesture-based Text Input Solution. In *Proceedings of AVI 2012, 11th International Working Conference on Advanced Visual Interfaces*, pages 156–163, Naples, Italy, May 2012.

[3] C.-M. Karat, C. Halverson, D. Horn, and J. Karat. Patterns of Entry and Correction in Large Vocabulary Continuous Speech Recognition Systems. In *Proceedings of CHI 1999, ACM Conference on Human Factors in Computing Systems*, pages 568–575, Pittsburgh, USA, May 1999.

[4] P. O. Kristensson, J. Clawson, M. Dunlop, P. Isokoski, B. Roark, K. Vertanen, A. Waller, and J. Wobbrock. Designing and Evaluating Text Entry Methods. In *Proceedings of CHI 2012, ACM Conference on Human Factors in Computing Systems*, pages 2747–2750, Austin, USA, May 2012.

[5] I. MacKenzie and 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.

[6] M. R. Morris. Web on the Wall: Insights From a Multimodal Interaction Elicitation Study. In *Proceedings of ITS 2012, International Conference on Interactive Tabletops and Surfaces*, pages 95–104, Cambridge, USA, November 2012.

[7] A. Schick, D. Morlock, C. Amma, T. Schultz, and R. Stiefelhagen. Vision-based Handwriting Recognition for Unrestricted Text Input in Mid-Air. In *Proceedings of ICMI 2012, 14th International Conference on Multimodal Interaction*, pages 217–220, Santa Monica, USA, October 2012.

[8] K. C. Sim. Speak-As-You-Swipe (SAYS): A Multimodal Interface Combining Speech and Gesture Keyboard Synchronously for Continuous Mobile Text Entry. In *Proceedings of ICMI 2012, 14th International Conference on Multimodal Interaction*, pages 555–560, Santa Monica, USA, October 2012.

[9] C. Szentgyorgyi and E. Lank. Five-Key Text Input Using Rhythmic Mappings. In *Proceedings of ICMI 2007, 9th International Conference on Multimodal Interfaces*, pages 118–121, Nagoya, Japan, November 2007.

[10] K. Vertanen and P. Kristensson. Parakeet: A Continuous Speech Recognition System for Mobile Touch-Screen Devices. In *Proceedings of IUI 2009, 14th International Conference on Intelligent User Interfaces*, pages 237–246, Sanibel Island, USA, February 2009.

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

[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] 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.

[14] J. Yuan, M. Liberman, and C. Cieri. Towards an Integrated Understanding of Speaking Rate in Conversation. In *Proceedings of Interspeech 2006, 9th International Conference on Spoken Language Processing*, pages 541–544, Pittsburgh, USA, September 2006.