Humsher: A Predictive Keyboard Operated by Humming

Ondřej Poláček Zdeněk Míkovec Adam J. Sporka Pavel Slavík
Faculty of Electrical Engineering, Czech Technical University in Prague,
Karlovo nám. 13, 12135 Praha 2, Czech Republic
{polacond, xmikovec, sporkaa, slavik}@fel.cvut.cz

ABSTRACT
This paper presents Humsher – a novel text entry method operated by the non-verbal vocal input, specifically the sound of humming. The method utilizes an adaptive language model for text prediction. Four different user interfaces are presented and compared. Three of them use dynamic layout in which n-grams of characters are presented to the user to choose from according to their probability in given context. The last interface utilizes static layout, in which the characters are displayed alphabetically and a modified binary search algorithm is used for an efficient selection of a character. All interfaces were compared and evaluated in a user study involving 17 able-bodied subjects. Case studies with four disabled people were also performed in order to validate the potential of the method for motor-impaired users. The average speed of the fastest interface was 14 characters per minute, while the fastest user reached 30 characters per minute. Disabled participants were able to type at 14 – 22 characters per minute after seven sessions.

Categories and Subject Descriptors
H.5.2 Information interfaces and presentation: User Interfaces – Input devices and strategies; Keyboard.

General Terms
Measurement, Performance, Design, Experimentation, Human Factors.

Keywords
Non-verbal Vocal Interface, Assistive Technology, Text Input, Predictive Keyboard, Adaptive Language Model

1. INTRODUCTION
Research in the field of text entry methods has been widely documented for some time. In static desktop environments we can observe the dominance of QWERTY keyboard which is caused by its extreme popularity rather than its optimal performance. Learning a new layout is a tedious process that can take more than 100 hours [1]. However, in special circumstances (e.g., impaired users, mobile environment) no dominant text entry method can be identified. This has consequently led to the development of many non-traditional approaches, where users accept longer learning time.

The maximum realistic text entry speed can be defined as a speed of an experienced typist using ten fingers on QWERTY keyboard. The speed will be approximately 250-400 characters per minute (CPM) for a professional typist [2]. With this speed achieved there is a little space for any enhancements like predictive completion, dynamic layouts, etc. as this will effectively slow down the type rate.

Physically disabled people usually cannot achieve such high speed due to their constraints. Their communication with computers is rather limited to only several distinctive stimuli – small number of physical buttons, joystick, eye-tracking, features of the electroencephalographic (EEG) signal etc. This limitation can be compared to a situation when we are typing with one finger only on virtual keyboard displayed on a touch screen. There is a research available [3], showing that typing with one finger on a touch screen with virtual QWERTY keyboard results in a speed 160 CPM for expert users after 30 minutes training. If we reduce the size of the virtual keyboard to 7 cm then the speed will drop to 105 CPM. The speed reached by physically disabled people will be certainly lower. This situation opens a space for research of new entry methods which will take into account various limitations of motor impaired users and increase the entry speed.

There is currently a range of assistive tools available to help users with motor impairments. However, each user may have significantly different capabilities and preferences according to the range and degree of their impairment. In case of severe physical impairment, people usually have to use other interaction methods to emulate the keyboard. One of the methods that has been successfully used by people with special needs is the non-verbal vocal interaction (NVVI) [4]. It can be described as an interaction modality, in which sounds other than speech are produced, for example humming [27] or vowels [28].

Our virtual keyboard, *Humsher*, described in this paper utilizes vocal gestures, i.e. short melodic and/or rhythmic patterns. The user can operate the keyboard by humming. Each key is assigned a pattern. It has been designed for those people with upper-limb motor impairments such as quadriplegia induced from stroke, cerebral palsy, brain injury etc. Additionally, users are required to have healthy vocal folds enough to be able to produce humming. The main advantages of such interaction are its language independence and fast and accurate recognition as opposed to speech [4]. Speech recognition software usually works relatively well for native speakers; however, the accuracy is much lower for accented speakers or for people with speech impairment.

1.1 Definitions of Terms
Probably the most common measures of performance of text entry methods are *words per minute* (WPM) or *characters per minute*

(CPM) [29]. Both rates indicate speed of a text entry method. Relation between them is defined by Equation 1. ISO 9241-4 standardizes WPM rate for keyboards at CPM divided by five, i.e. one "word" is considered as five characters including spaces. CPM is defined by equation 2, where |T| is length of written text in characters and S is time in seconds.

$$WPM = \frac{1}{5} \times CPM. \quad \textbf{(1)}$$

$$CPM = \frac{|T| - 1}{S} \times 60. \quad \textbf{(2)}$$

A *gestures per character* (GPC) rate [29] is also used in this paper for evaluating purposes. Gesture is regarded as an atomic operation. In the case of the humming input, vocal gestures are treated as atomic operations. Text entry methods with low GPC rate are considered as better than those with high rate; however, other parameters must be taken into account, such as length or complexity of the vocal gesture. The GPC rate is defined by Equation 3, where $|IS_{\emptyset}|$ is an input stream which contains all vocal gestures produced by the user and |T| is length of written text in characters.

$$GPC = \frac{|IS_{\emptyset}|}{|T|}. \quad \textbf{(3)}$$

A sequence of *n* characters is referred to as *n-gram*. The n-grams with length equal to one, two and three character are being called unigrams, bigrams and trigrams respectively. In the paper the term *n*-gram is used for strings of characters of an unspecified length *n*.

2. RELATED WORK

There is a wide range of text entry methods targeting the motor-impaired users. We can notice that the methods described in this section often differ significantly in physical interaction used, which is determined by specific motor impairment. Each method is often unique for concrete impairment conditions and thus it typically makes no sense to compare various methods as they are not in concurrent position. Several principles can be identified in the literature – *predictive completion*, *ambiguous keyboards* and *scanning*.

A text entry method can be accelerated by prediction, when a list of possible completions is updated with each entered character. This reduces number of keystrokes per character. The Reactive Keyboard [5] predicted possible words according to context that had been already written. An adaptive dictionary-based language model was used. Predicted candidates could be selected by the mouse cursor. Expert users of a QWERTY keyboard would be slowed down, however, such prediction is useful for poor typist or people with limited movement of upper limbs. Another predictive keyboard GazeTalk [6] predicted six most probable letters and six words according to current context. If no prediction was correct, there was full keyboard available. This virtual keyboard was controlled by the eye gaze. The keys were activated by dwell-time selection system [7]. The average typing rate achieved by novice users was 16 CPM.

Probably the most prevalent ambiguous keyboard is the commercial T9 system by Tegic Communications [8] that is widely adopted on mobile phones. The idea behind is simple – the alphabet is divided into nine groups of characters and then each group is assigned to one key. The user selects desired characters by selecting the keys and after a sequence of keys is entered the word is disambiguated using a dictionary. For its efficiency, similar ambiguous keyboards were designed for physically impaired people. Kushler [9] describes an ambiguous keyboard in which the alphabet was assigned to seven keys and the eighth key was used as a space key that initiated the disambiguation process. Tanaka-Ishii [10] published similar system, in which only four physical keys were used. Besides disambiguation, the text entry method was capable of predicting words. The average speed of this method was 70 CPM, achieved after ten sessions by able-bodied participants. Harbusch [11] presented similar method in which the whole alphabet was assigned to only three keys and one key was used for executing special command in a menu.

When the number of stimuli, which can be issued by the user, is limited to only one or two, using scanning technique is inevitable. For example in the case of two buttons, the first button can be pressed repetitively (scanning) to select a key and second button is used to confirm the selection. When only one button is available, the keys are selected automatically for a certain amount of time. After the time expires, next key is selected. The button is used to confirm the selection again. Keys can be spatially organized in a matrix and the desired key is then selected by row-column scanning [12]. Combining linear scanning with an ambiguous keyboard is a common technique. For example, Kühn [13] used four-key scanning ambiguous keyboard and achieved 35 CPM without out-of-vocabulary words. Miro [14] limited the number of keys to only two (keys 'a-m' and 'n-z') and estimated its entry rate to 50 CPM for an expert user. Beltar [15] used three keys and developed a virtual mobile keyboard. In QANTI [16] three keys are mapped to the alphabet. The keyboard is operated by one switch that is triggered by intentional muscle contractions. The typing rate ranges from 12.5 to 33 CPM.

An efficient system is Dasher [17], which is based on a dynamically modified display and adaptive language model [18]. The characters are selected by moving the mouse cursor around the screen. Continuous "one finger" gestures are used as the input method. This is a very suitable input method for motor impaired users, who can operate a pointing device. The writing speed achieved is approximately 100 CPM with experienced users reaching up to 170 CPM. For users who have no hand function, a modification of the Dasher system can be made to allow input via eye tracking. A longitudinal study [19] found that an average writing speed of 87 CPM after ten 15-minutes sessions could be achieved. This speed was a large increase from the initial speed of just 12.5 CPM. Speech Dasher [20] is another interesting modification of Dasher. It combines speech input with the zooming input of Dasher. The system must first recognize a user's utterance. Errors are then corrected via the zooming input. Expert users reached a writing speed of approximately 200 CPM.

Sporka et al. [21] describe the NVVI-based method of keyboard emulation. Each vocal gesture is assigned a specific key on the keyboard, when a gesture is produced a corresponding key is emulated. The average reported typing rates varied between 12 and 16 CPM, which was measured in a study with able-bodied participants. Different assignments of NVVI gestures to keys were investigated, namely the pitch-to-address, pattern-to-key and Morse code mappings. In the pitch-to-address mapping, the keyboard was mapped onto a 4×4×4 matrix, while a sequence of three tones of specific pitches determined an address in the matrix. In the pattern-to-key mapping each key was assigned a specific gesture.

Another keyboard operated by NVVI is CHANTI [25]. It is an ambiguous keyboard, where the alphabet is split into only three groups. The keyboard combines philosophy of the ambiguous keyboard QANTI [16] and humming input. Scanning technique is replaced by direct selection of a key by vocal gestures. The keyboard was tested with five severely motor-impaired people, the speeds ranged from 10 to 15 CPM after 7 sessions.

Additionally, the P300 speller [22] is a method that utilizes the electroencephalographic (EEG) signal in the human brain to control a virtual keyboard. The keyboard is a 6x6 matrix containing alphanumeric characters. The user focuses on a character and as the character flashes, the brain produces a stimulus. At least two flashes are needed to input a character. According to Wang et al. [22], the writing speed achieved is approximately 7.5 CPM.

3. HUMSHER DESIGN

Our virtual keyboard, *Humsher*, has been designed for severely motor-impaired people, who can control it by vocal gestures. It utilizes the same language model as Dasher [17] (prediction by partial match; PPM [18]). The model provides n-grams and their probability, which have been predetermined by a given context. The model is initialized from a small corpus of English text, but it adapts as the user types.

3.1 Dynamic Layouts

The interfaces described in this section employ dynamic layout. The n-grams, which are extracted from the PPM model, are offered sorted according to their probability. The probability is predetermined by already written text. Practically it means that after typing an n-gram, the context is updated, probabilities of following n-grams are recounted and the layout is displayed accordingly.

We designed and implemented three different user interfaces (Direct, Matrix and List) with dynamic layout of characters. Each interface differs in either vocal gesture set or in mapping of gestures to actions. The Direct and Matrix interfaces utilize six vocal gestures as depicted in Fig. 1, whilst the List interface utilizes only three simple vocal gestures as depicted in Fig. 2. The vocal gestures are explicitly identified by its length (short/long) or by its pitch (low/high). In order to distinguish low and high tones a threshold pitch needs to be adjusted for each user – e.g. the difference between male and female voice is as much as one or two octaves. Only two different pitches were chosen as with increasing number of pitches, more precise intonation is required and the interaction becomes more error prone.

All three interfaces offer n-grams, containing the characters how the text might continue, sorted according to the probability. The n-grams can be unigrams (individual characters) as well as bigrams, trigrams, etc. The length of n-grams is not limited, only probability matters. N-grams to display are chosen according to the following steps:

1. Add all unigrams to the list L that will be displayed.
2. For each n-gram in the list L compute probability of all (n+1)-grams and add them to the list L if their probability is higher than a threshold.
3. Repeat step 2 until no n-gram can be added.
4. Sort the list L according to probability of each n-gram.

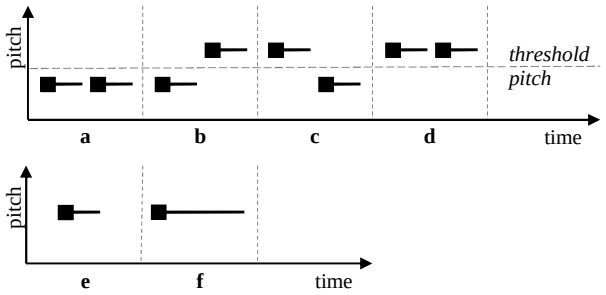

Figure 1. Vocal gestures used in Direct and Matrix interfaces

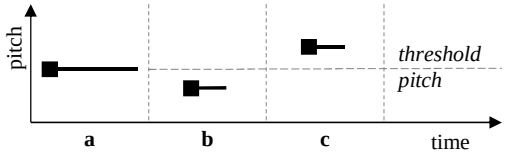

Figure 2. Vocal gestures used in List and Binary interfaces

3.1.1 Direct interface

The *Direct* interface (see Fig. 3) allows users to directly choose from four cells (labeled cell 1 to 4) in the Active column (part A). These cells contain n-grams that have been determined as the most probable following characters of the written text. Cells can be selected by vocal gestures depicted in Fig. 1a-d:

a. two consequent low tones (cell 1),
b. a low tone followed by a high tone (cell 2),
c. a high tone followed by a low tone (cell 3),
d. two consequent high tones (cell 4).

If there is no cell in the Active column that contains the desired character, the user has to move the leftmost column in the Look ahead (part B) to the Active column by producing a single short tone (see Fig. 1e) and keep repeating it until the desired n-gram appears in one of the cells in Active column. Text, which has been already written, can be erased by producing a long tone (see Fig. 1f). The longer the user keeps producing the tone the faster are the characters erased.

3.1.2 Matrix interface

The *Matrix* interface (see Fig. 4) utilizes the same vocal gestures as the Direct interface, however, the user interaction is different. Users are presented with a 4×4 matrix of the most probable n-grams. Cells in the left column of the matrix contain the highest probable n-grams, whilst the rightmost cells contain the lowest probable n-grams.

Selection of the correct cell is accomplished in two steps by specifying a column and a row. First, the user must select a column by producing a corresponding vocal gesture (Fig. 1a-d). The column is then highlighted and the same vocal gestures can be used to select the desired cell by selecting a row. If a character does not appear in the matrix, the user has to produce a short tone (see Fig. 1e) in order to display less probable n-grams. Written text can be erased by producing a long tone (see Fig. 1f), in the same manner as in the Direct interface.

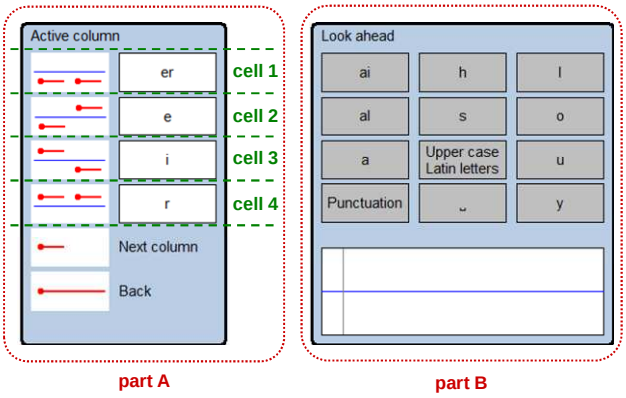

Figure 3. Direct interface. A– active column, C B – look ahead matrix

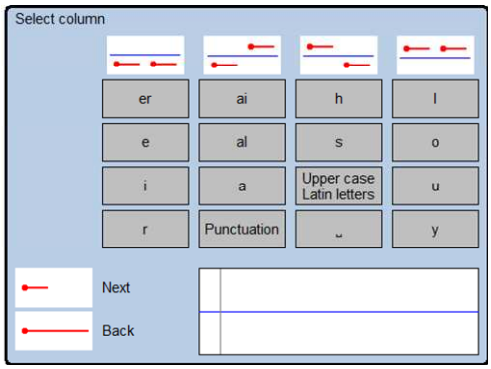

Figure 4. Matrix interface

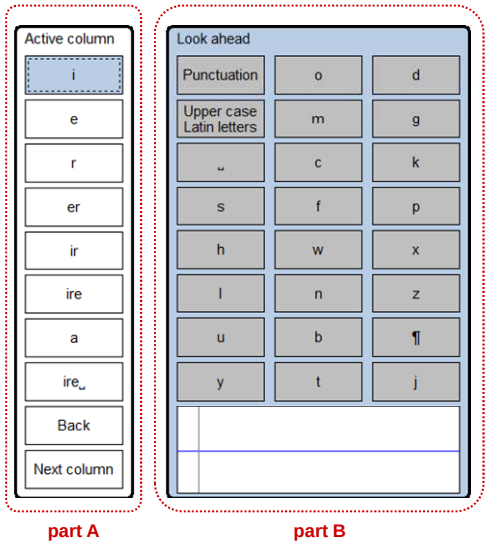

Figure 5. List interface. A – active column, B – look ahead matrix

3.1.3 List interface

The *List* interface (see Fig. 5) is controlled by just three simple and easy-to-learn gestures (see Fig. 2). The Active column (part A) presents the user a list of cells containing the eight most probable n-grams. The topmost cell is selected. Users can move the selection up and down by producing a short high or low tone (see Fig. 2b,c). A long tone (see Fig. 2a) is used to confirm the desired selection. This interface does not utilize special vocal gestures to select the next column or erase written text. Instead, these two functions are always made available by introducing two special cells Back and Next column at the bottom of the Active column list.

3.2 Static Layout

Static layout was designed in order to simplify the process of visual location of desired character. In dynamic layouts users have to locate a character visually by linear scanning and they cannot rely on the visual memory. The process of locating correct character can be tedious for low-probable characters. Moreover, users sometimes do not notice a correct character and they have to rotate through the whole list of characters and n-grams once again. This consequently can lead to users' frustration. Therefore we decided to implement a static interface, called Binary interface, that keeps position of characters and the characters are sorted alphabetically. Time needed to locate a character is then modeled by Hick-Hyman law [24] and it is logarithmically dependent on the length of the alphabet. Locating characters visually in the static layout is obviously faster than the same task in dynamic layouts as logarithmic scanning is used instead of linear.

3.2.1 Binary interface

In the *Binary* interface (see Fig. 6) the characters are always displayed in an alphabetic order. Such order gives us an opportunity to select desired character by binary search algorithm adopted from basic programming techniques. The algorithm locates position of a character in the alphabet by splitting it into two halves and deciding which half is used in the next step. Then the half is split again and again until the correct character is found. Each character is located in following number of steps:

$$steps = \lceil \log_2 N \rceil \qquad \textbf{(4)}$$

N is size of the alphabet. In our case the algorithm would require $\lceil \log_2 36 \rceil = 6$ selections as our alphabet contains 36 symbols. The user would have to produce six vocal gestures to enter a character. Therefore the best theoretical GPC rate achieved by the binary search is equal to six, which is quite high. But what happens if the alphabet is split according to the probability of characters rather than into two exact halves? Then a character with high probability could be located in fewer steps, however, character with low probability might be located in even more than six steps. The actual GPC rate measured empirically in a user study presented later is much lower than six.

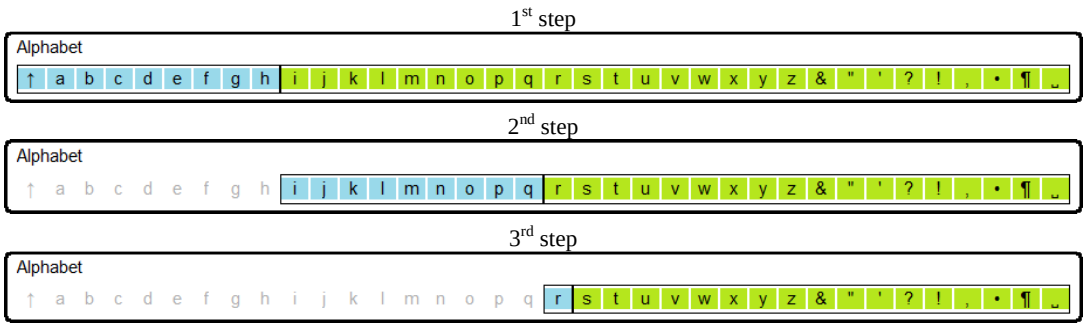

Figure 6. Binary interface, typing "*r*" after "*Text ent*"

The Binary interface is based on modified binary search algorithm. In each step the alphabet is split into two groups with balanced probability, i.e. the sum of probabilities of characters in each group is as close to 0.5 as possible. The boundary between groups is then computed according to the Equation 5, where k is the index of boundary character, p_i is a probability of character i and N is a size of the alphabet.

$$\min_{1<k<N} \left(\left(\frac{1}{2} - \sum_{i=1}^{k-1} p_i \right)^2 + \left(\frac{1}{2} - \sum_{i=k}^{N} p_i \right)^2 \right). \qquad (5)$$

The Binary interface utilizes only three vocal gestures (see Fig. 2) as well as the List. Short low tone (Fig. 2b) and short high tone (Fig. 2c) are used for entering text, while the long tone (Fig 2a) is used for corrections.

An example of user interaction with the Binary interface is depicted in Fig. 6. Let us assume that the user has already entered the text "Text ent" and wants to continue by entering character "r". In the first step the alphabet is split into two groups "shift -h" and "i-space". The user chooses the second group by producing a high short tone. In the second step the rest of the alphabet is split into groups "i-q" and "r-space". Again the second group is chosen by the same high short tone. In the last step "r" is the only character in the first group because of its high probability. Remaining characters are in the second group. The character "r" is now entered by low short tone. In this case the character was selected only in three steps by three short tones.

When comparing Binary interface to the other three interfaces, several features can be observed:

- User can easily locate desired character as letters are sorted alphabetically and characters do not change their positions while entering text.
- Simple vocal gestures are employed (similar to List interface). Only two gestures are used for entering text and one for deleting text.
- The Binary interface offers only single characters unlike the interfaces with dynamic layout. It is not possible to enter more characters at once.

4. EVALUATION

In order to evaluate the interfaces we conducted two user studies. The goal of the first one was to compare all four interfaces, measure their speed and find out user's opinions on them. In the second study four disabled participants were recruited to validate potential of Humsher for motor-impaired users.

4.1 Comparison of interfaces

The aim of the user study was to measure the writing speed of each interface and subsequently determine which interface was the most efficient. In the study 17 able-bodied participants (10 men, 7 women, mean age=26, SD=2.1) took part. Each participant completed four sessions. According to Mahmud et al. [23], four sessions are needed to minimize the error rate of the NVVI. The schedules of each session are outlined below:

- **Session 1:** Participants were trained in producing the required vocal gestures. After reaching an accuracy of 90%, they were presented with all interfaces and asked to enter short phrases with each of them. This session lasted approximately 30-60 minutes depending on the user's abilities.
- **Sessions 2 and 3:** Participants were asked to enter two simple phrases using all interfaces. The sessions were conducted remotely and they lasted roughly 20 minutes.
- **Session 4:** Participants were asked to enter three phrases using all interfaces. The session was conducted remotely and it lasted roughly 30 minutes. Objective data from this session were collected.

After the last session each participant performed a subjective evaluation of each interface by means of remote interview. The participants received approximately 24 hours rest between the sessions. In order to minimize the learning effect, the sequence of interfaces was counterbalanced. Objective results (CPM, GPC rate and number of corrections) are shown in Table 1.

Table 1. Means and standard deviations (SD) of the typing rate (CPM), vocal gesture per character (GPC) rate and total number of corrections.

Interface	CPM		GPC		Corrections	
	Mean	SD	Mean	SD	Mean	SD
Direct	14.4	2.8	1.8	0.23	13.0	11.0
Matrix	11.8	2.1	1.9	0.32	16.1	14.6
List	13.0	3.2	3.5	0.58	6.4	6.6
Binary	11.7	1.8	3.4	0.18	14.5	8.5

The ANOVA test and Scheffé's method [26] were used to find statistically significant differences in mean quantities among interfaces. When comparing mean CPM rates, the Direct interface was significantly faster ($F(3,67) = 4.20$, $p < .01$) than the Matrix interface and it was also significantly faster than the Binary interface. Other differences in speed were not significant.

In the case of List and Binary interfaces, the users had to produce significantly more ($F(3,67)= 107.7$, $p < .01$) vocal gestures per character than Direct and Matrix interfaces. This corresponds to number of vocal gestures used in the interfaces. Direct and Matrix interfaces utilize six complex gestures (see Fig. 1), while the other interfaces only three simple gestures (see Fig. 2). As mentioned in section 3.2.1, theoretical GPC rate for standard binary search is 6, when the alphabet contains 36 symbols. By modifying the binary search, we succeeded to reduce the GPC rate to 3.4 empirically measured in the user study.

After the last session, participants were asked to comment on the interfaces. The Direct interface was mostly perceived as accurate and fast. The Matrix interface was in many cases perceived as fastest among all interfaces, although it was slower than Direct and List interfaces. Additionally, the List interface, which is not the slowest, was reported as the slowest. The List interface was also reported as cumbersome – some participants complained that it was not transparent enough and the navigation was tedious. This is probably due to the high number of cells in columns, which makes the visual searching more difficult. The Binary interface was found easy and fast by most participants, although it was the slowest one. The participants appreciated static layout of the interface, however, eight participants complained about the fact that only one character can be entered at one time and the method does not offer n-grams as the dynamic layout interfaces. The participants also made positive comments on simplicity of vocal gestures used to control the interface. Although there were no significant differences in objective data between List and Binary interfaces, participants strongly preferred the Binary one.

We identified two main searching strategies employed by participants when using Direct and List interfaces. Some of them visually scanned only the first column (Active column, see Fig. 3 and 4). When searched character was not found in this column, they moved forward and scanned the first column again. Some of them also reported that the Look ahead matrix is redundant and confusing. The other participants visually scanned all cells in Active column and Look ahead matrix. When searched character was not found, they moved forward and scanned the last column. They reported that this strategy allows them to plan vocal gestures in advance, which they found faster.

Ten participants reported fatigue of vocal folds during the experiment, which they mostly compensated for by lowering their pitch and dropping their voice.

Table 2. Performance of expert users

Interface	Expert 1			Expert 2			Expert 3		
	CPM	GPC	corr	CPM	GPC	corr	CPM	GPC	corr
Direct	29	1.5	1	24	1.7	8	30	1.5	2
Matrix	23	1.6	3	20	1.9	15	23	1.5	2
List	25	2.8	0	17	3.4	4	26	2.9	1
Binary	23	3.6	1	16	3.6	23	20	3.2	10

4.1.1 Typing rate of expert users

Learning a new text entry method is always a long-term process. The study presented results of novice users, who were given only necessary amount of training. In order to determine possible upper limit of performance of all Humsher interfaces, three experienced NVVI users were given 4-6 hours of training. The typing rate was recorded after their performance did not improve significantly. Table 2 summarizes CPM, GPC rates and number of corrections for each interface. The speed varied between 16 and 30 CPM. Expert 1 and 3 preferred the Direct, while expert 2 preferred Matrix interface.

4.2 Case studies with disabled people

The goal of the study was to find out whether Humsher can serve as an assistive tool for motor-impaired people. Four people were recruited in cooperation with local non-profit associations. The study was longitudinal, it was organized in seven sessions and each session lasted 30-60 minutes. First, the participants were asked to use the Binary interface because of its simple vocal gestures. Then they were asked to learn more complicated gestures and use the Direct interface, because it was the fastest one. The rough schedules of each session are outlined below:

- **Session 1:** The participants were asked to describe how they use ICT and how they enter text. Then they were trained in producing vocal gestures starting with the easiest ones (see Fig. 2). Binary interface was presented and the participants were asked to enter a phrase.
- **Session 2:** Participants trained more complicated vocal gestures (see Fig.1) until required accuracy was achieved. Then the Direct interface was presented to them and they were asked to enter a few phrases.
- **Session 3 – 7:** Participants were asked to enter phrases using the Direct interface. On the last day the participant were asked to describe experience using the interfaces.

While training the vocal gestures, the thresholds for low/high and short/long tones were personalized for each user. Two users with speech impairments were not able to consciously alter pitch of their tone, therefore a new gestures were designed especially for them.

4.2.1 Participant 1

The participant was 30 year old IT specialist in a small company, quadriplegic since birth. Due to privacy protection, he only participated in the study remotely. We conducted interviews with him via telephone and e-mail.

He uses a mouth stick to operate his PC (keyboard and mouse). Apart from the Sticky Keys tool available in Microsoft Windows he uses no other assistive technology. He uses various system administration tools, word processors, graphic and sound editors and he feels no disadvantage in comparison with other users.

He found the Direct interface precise and pleasant to use. Overall, he said he felt in control when using the tool. "The system allowed me to write whatever I wanted. I was not forced into any options." He used the word "intelligent" to describe the suggested options provided by the tool when typing text. He achieved a mean type rate of 22 CPM. He reported, however, that his current text entry rate achieved by the mouth stick is higher.

4.2.2 Participant 2

Another disabled participant was 19 years old, quadriplegic since an accident about 3 years ago. He is a high-school student who uses computer to access study materials, talk with his friends over text media (especially e-mails), make telephone calls and watch movies. He spends typically 2 to 4 hours using his laptop equipped with NaturalPoint SmartNav4 head motion tracker and Click-N-Type keyboard emulation software. However, he is able to use the head motion tracking system only for 2-4 hours and then he gets too tired. He had a previous experience with another NVVI based interface for entering text.

When working with Binary interface, his mean type rate was 12 CPM. After switching to Direct interface, the type rate increased to 21 CPM. Although he was almost two times faster with the Direct interface, he reported that the Binary interface was quicker and more responsive ("I like that it is fast. I can see it all in front of me and I know exactly what to do next."). He felt more in control than when using the Direct interface ("I am a bit lost when using the Direct interface as I sometimes do not notice the right option."). The participant considered our method similar in speed to his current assistive technology and he would use it as an alternative solution when his head gets too tired.

4.2.3 Participant 3

The participant was a 58 year old woman with cerebral palsy. All her limbs are affected by the disease. She can sit on a chair, but she needs a wheelchair for movement. She has a lot of unintentional movements in her arms. Her voice is also affected. She speaks slowly and she does not articulate properly. Her health state is slowly but steadily declining.

She used to work as an office staff in a non-profit organization, but she is unemployed for one year now. She used to type on a typewriter and a computer keyboard. However, now her performance decreases and she types very slowly on a keyboard. The only assistive technology that she uses is a trackball to control the mouse pointer. She also tried speech recognition, but it did not work for her at all.

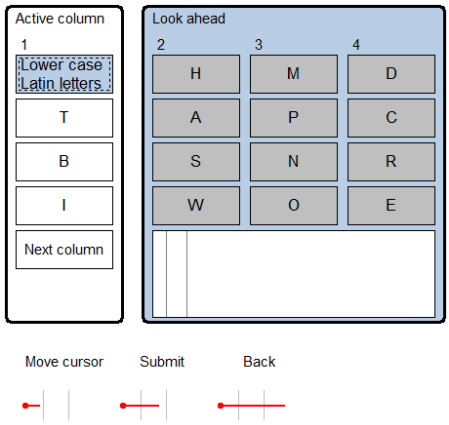

Figure 7. Modified List interface

She spent first and second sessions trying to learn voice gestures for the Binary interface. However, after two sessions she could hardly write a phrase. She was not able to effectively alter pitch of her tone, which led to many corrections. Therefore the vocal gestures were changed to short, medium long and long tone. Then she was asked to use it for another two sessions and she reached 8 CPM.

As the participant was unable to produce more complicated gestures, we modified the List interface (see Fig. 7) for use with the new gesture set. Short tone was used to move cursor in the Active column down, medium tone to submit selected n-gram and long tone for correction. She used this interface for remaining three sessions and reached 15 CPM.

The participant reported that the speed of the modified List interface is similar to her current typing rate and she was interested in purchasing it as a product. She also made comments on speech recognition ("This is much better than speech for me"). She reported that after one hour of humming her vocal chords were not tired at all.

4.2.4 Participant 4

The participant was 51 years old, quadriplegic since an accident about 22 years ago. His legs and right arm are paralyzed. He can use his left arm to operate wheelchair, however, fine motoric of his left hand is reduced. His vocal chords and neck muscles are also slightly affected.

Before the accident he used to work as a machine engineer. Since that he is unemployed. He has never worked with computers, but he regularly uses cell phone for couple of years, mainly for calling and writing short text messages. However, composing message is a tedious process for him.

The participant started with Binary interface and used it for two sessions. He experienced similar problems to participant 3. As he was not able to produce low and high tone properly, his performance was about 1 CPM with a lot of corrections. In the third session he switched to the modified List interface (see Fig. 7) as participant 3 and his performance increased rapidly with minimum mistakes. Using this interface and the vocal gestures based on length he reached type rate of 14 CPM.

He stated that typing text with Humsher is faster and better than typing on his cell phone. Generally he was pleased with the modified List interface. However, his vocal chords got tired after 40 minutes of humming.

5. CONCLUSION

This paper has presented and evaluated four interfaces of Humsher – an adaptive virtual keyboard operated by humming. Three of them (Direct, Matrix and List) used dynamic layout, in which characters were sorted according to its probability. The layout was updated after entering a character. The last interface (Binary) used a static layout, in which characters were displayed alphabetically and did not change their position. A character was selected by modified binary search algorithm that took into account probability of each character.

Most novice users preferred the Binary interface, even though it was not the fastest one. They appreciated mostly the static layout of characters and simple vocal gestures used to control the interface. On the other hand expert users preferred interfaces with dynamic layouts. Interfaces with dynamic layout were perceived worse, however, users appreciated that sometimes several characters could be entered together. The Direct interface was the fastest one with average speed 14.4 CPM achieved by novice and 28 CPM by expert users.

Acceptance of our tool for the target group was verified by the inclusion of four motor-impaired participants. Two of them could not use speech recognition software as their speech was also impaired. Cases of all disabled participants are described separately in a longitudinal and qualitative study. Their speed achieved after seven sessions varied between 14 and 22 CPM.

While some techniques, such as Dasher [19], offer their users type rates up to 100 CPM, they may not be used by people with severe motor impairments without expensive hardware, such as eye trackers. Our method requires no additional hardware to a standard PC and performs better than the NVVI Keyboard [21] and CHANTI [25] methods which have the identical hardware requirements and for which a similar performance is reported: 16

CPM for NVVI Keyboard, 15 CPM for CHANTI, and 22 CPM for Humsher.

6. ACKNOWLEDGMENTS

This research has been supported by the MSMT research program MSM 6840770014 and the Veritas project (IST-247765).

7. REFERENCES

[1] Silfverberg, M. 2007. Historical Overview of Consumer Text Entry Technologies. In Text Entry Systems: Mobility, Accessibility, Universality, I. S. MacKenzie and K.Tanaka-Ishii (eds.). Morgan Kaufmann, 3-26.

[2] West, L. J. 1998. The Standard and Dvorak Keyboards Revisited: Direct Measures of Speed. http://samoa.santafe.edu/media/workingpapers/98-05-041.pdf, Technical report, Santa Fe Institute.

[3] Sears, A., Revis D., Swatski, J., Crittenden, R., Shneiderman, B. 1993. Investigating touchscreen typing: the effect of keyboard size on typing speed. In J. Behaviour and Information Technology, vol. 12, 17-22.

[4] Igarashi, T., Hughes, J F. 2001. Voice as sound: using non-verbal voice input for interactive control. In Proceedings of UIST ’01, ACM Press, 155-156.

[5] Darragh, J. J., Witten, I. H., James, M. L. 1990. The Reactive Keyboard: A Predictive Typing Aid. Computer Journal, vol. 23, IEEE Press, 41-49.

[6] Hansen, J., Johansen, A., Hansen, D., Itoh, K., Mashino, S. 2003. Language technology in a predictive, restricted on-screen keyboard with ambiguous layout for severely disabled people. In Proceedings of EACL Workshop on Language Modeling for Text Entry Methods.

[7] Jacob, R. J. K. 1990. What you look at is what you get: eye movement-based interaction techniques. In Proceedings of CHI’90, ACM Press, 11-18.

[8] Grover, D. L., King, M. T., Kushler, C. A. 1998. Reduced keyboard disambiguating computer, Technical report, US Patent Publication.

[9] Kushler, C. 1998. AAC: Using a Reduced Keyboard.

[10] Tanaka-Ishii, K., Inutsuka, Y., Takeichi, M. 2002. Entering text with a four-button device. In Proceedings of the 19th International Conference on Computational Linguistics, Association for Computational Linguistics, 1-7.

[11] Harbusch, K., Kühn, M. 2003. Towards an adaptive communication aid with text input from ambiguous keyboards. In: Proceedings of EACL’03, Association for Computational Linguistics, 207-210.

[12] Simpson, R., Koester, H. 1999. Adaptive one-switch row-column scanning. In IEEE Transactions on Rehabilitation Engineering, vol. 7, no. 4, IEEE Press, 464-473.

[13] Kühn, M., Garbe, J. 2001. Predictive and highly ambiguous typing for a severely speech and motion impaired user. In Proceedings of 1st International Universal Access in Human-Computer Interaction Conference, UAHCI ’01.

[14] Miro, J., Bernabeu, P. 2008. Text entry system based on a minimal scan matrix for severely physically handicapped people. In Computers Helping People with Special Needs, LNCS 5105, Springer, Heidelberg, 1216-1219.

[15] Belatar, M., Poirier, F. 2008. Text entry for mobile devices and users with severe motor impairments: handiglyph, a primitive shapes based onscreen keyboard. In Proceedings of ASSETS ’08, ACM Press, 209-216.

[16] Felzer, T., MacKenzie, I., Beckerle, P., Rinderknecht, S. 2010. Qanti: A software tool for quick ambiguous non-standard text input. In Computers Helping People with Special Needs, LNCS 6180, Springer, 128-135.

[17] Ward, D. J., Blackwell, A. F., MacKay, D. J. C. 2000. Dasher – a data entry interface using continuous gestures and language models. In Proc. of UIST ’00, ACM, 129-137.

[18] Teahan, W. 1995. Probability estimation for PPM. In: Proceedings of the New Zealand Computer Science Research Students’ Conference.

[19] Tuisku, O., Majaranta, P., Isokoski, P., Räihä, K. J. 2008. Now Dasher! dash away!: Longitudinal study of fast text entry by eye gaze. In Proceedings of the 2008 symposium on Eye tracking research and applications, ETRA ’08, ACM Press, 19-26.

[20] Vertanen, K., MacKay, D. J. 2010. Speech dasher: fast writing using speech and gaze. In Proceedings of CHI ’10, ACM Press, 595-598.

[21] Sporka, A. J., Kurniawan, S. H., Slavík, P. 2006. Non-speech operated emulation of keyboard. In: Clarkson, J., Langdon, P., and Robinson, P. (eds.) Designing Accessible Technology, Springer, London, 145-154.

[22] Wang, C., Guan, C., Zhang, H. 2005. P300 brain-computer interface design for communication and control applications. In: Proceedings of the 27th Annual International Conference of the Engineering in Medicine and Biology Society, IEEE-EMBS’05, 5400-5403.

[23] Mahmud, M., Sporka, A. J., Kurniawan, S. H., Slavik, P. 2007. A Comparative Longitudinal Study of Non-verbal Mouse Pointer, In Proceedings of INTERACT 2007, Springer, Heidelberg, 489-502.

[24] Hyman, R. 1953. Stimulus Information as a Determinant of Reaction Time. Journal of Experimental Psychology, vol. 45, 188-196.

[25] Sporka, A. J., Felzer, T., Kurniawan, S.H., Polacek, O., Haiduk, P., MacKenzie, I.S. 2011. CHANTI: Predictive Text Entry Using Non-verbal Vocal Input. In proceedings of CHI’11, ACM Press., 2463-2472.

[26] Maxwell, S.E., Delaney, H.D. 2004. Designing Experiments and Analyzing Data: A Model Comparison, ISBN 0805837183, Lawrence Erlbaum Associates, 217–218.

[27] Sporka, A.J., Kurniawan, S. H., Slavik, P. 2004. Whistling user interface (u3i) In 8th ERCIM International Workshop “User Interfaces For All”, LCNS 3196, Springer, 472-478.

[28] Harada, S., Landay, J.A., Malkin, J., Li, X. and Bilmes, J.A. 2006. The Vocal Joystick: Evaluation of voice-based cursor control techniques. Proceedings of ASSETS '06,ACM Press, 197-204.

[29] Wobbrock, J.O. 2007. Measures of text entry performance. In Text Entry Systems: Mobility, Accessibility, Universality, I. S. MacKenzie and K.Tanaka-Ishii (eds.). Morgan Kaufmann, 47-74.