Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Graphics and Interaction

Designing Text Entry Methods for Non-Verbal Vocal Input

by

Ing. Ondřej Poláček

A thesis submitted to
the Faculty of Electrical Engineering, Czech Technical University in Prague,
in partial fulfilment of the requirements for the degree of Doctor.

PhD programme: Electrical Engineering and Information Technology
Branch of study: Information Science and Computer Engineering

June 2014

Thesis Supervisor:
prof. Ing. Pavel Slavík, CSc.
Department of Computer Graphics and Interaction
Faculty of Electrical Engineering
Czech Technical University in Prague
Karlovo nám. 13
121 35 Praha 2
Czech Republic

Thesis Co-Supervisor:
Ing. Zdeněk Míkovec, Ph.D.
Department of Computer Graphics and Interaction
Faculty of Electrical Engineering
Czech Technical University in Prague
Karlovo nám. 13
121 35 Prague 2
Czech Republic

Copyright © 2014 by Ing. Ondřej Poláček

Abstract and contributions

In recent years, a lot of research effort can be observed within the field of computer accessibility focusing on design of text entry methods for people with various impairments. However, only little work have been devoted to non-verbal vocal input (NVVI), which is an interaction modality suitable for motor-impaired people.

The main goal of this thesis is to study how non-verbal vocal input (NVVI) can be applied to novel text entry methods and techniques in order to improve quality of life of motor-impaired people. This goal involves several aspects that needs to be explored in order to build a full picture of the problem. Solving each aspect brings us towards better understanding of the technology and the target users. Namely, this thesis focuses on applicability, acceptability, and accuracy of NVVI, combination of text input and NVVI, and text input optimization.

Main contributions of the thesis are the following:

1. A novel predictive text entry method based on n-grams and its evaluation with the target group.
2. Evaluation of an ambiguous text entry method with the target group.
3. Two novel text entry methods based on scanning.
4. A novel method for real-time segmentation of speech and non-verbal vocal input.
5. Subjective comparison of non-verbal vocal commands.

The thesis identifies the threshold of applicability of NVVI, defines its optimal target group, and shows how people with disabilities can be included by proper interaction design. The thesis also proposes guidelines, which offer NVVI designers a set of recommendations on how non-verbal vocal commands can be used efficiently. A novel method for speech and NVVI segmentation has been developed in order to improve accuracy of the input and filter spontaneous utterances. Several novel text entry methods have been designed and studied in the thesis. They improve the text input in specific contexts and surpass their predecessors in terms of entry rate or subjective perception. Models of the designed text entry methods have been proposed in order to optimize some of parameters that influence their performance. The work described in the thesis improves understanding of the areas of non-verbal vocal input and text entry methods. Partial results of the thesis were published in peer-reviewed journals and at international conferences.

Acknowledgements

First of all, I would like to express my gratitude to my thesis supervisor, prof. Pavel Slavík for his mentoring throughout my doctoral studies, valuable advices, and for taking care of my financial support. I would also like to thank Dr. Adam J. Sporka and Dr. Zdeněk Míkovec for consulting the research directions and helping me with design of experiments described in the thesis.

My thanks also go to Dr. Thomas Grill for his support during my sabbatical leave at the University of Salzburg. I would also like to thank all co-authors for their support and constructive criticism. Furthermore, I would like to express my thanks to all people who participated in studies and experiments.

Finally, my greatest thanks to my family and friends whose support was of great importance during my doctoral studies.

My work has been partially supported by MSMT under the research program MSM 6840770014, EC funded projects VitalMind (ICT-215387) and Veritas (IST-247765), project TextAble (LH12070; MSMT Czech Republic), SGS grant Automatically Generated UIs in Nomadic Applications (SGS10/290/OHK3/3T/13; FIS 10-802900), and grants FR-TI2/128 and TE01020415.

Contents

List of Figures

List of Tables

1 Introduction

Entering text is an important use case of many electronic devices including computers, mobile phones, tablets, etc. It is a challenging part of interaction between human and computer because of its complexity. For example, learning to type on a computer keyboard takes several months to achieve a reasonable entry rate. Entering text is then especially challenging for people who struggle with common interaction methods due to their impairments.

This thesis focuses on severely motor-impaired people who can use their non-verbal voice to interact with the computer. Based on this constraint, number of text entry methods are designed and evaluated. Text entry methods are used as a mean to enter characters into a computer. All of the text entry methods described in this thesis use voice as a mean of interaction. The thesis thus connects two fields within human-computer interaction (HCI): text entry methods and non-verbal vocal input. While text entry methods have been researched for quite a long time, the non-verbal vocal input is a relatively new interaction modality which appeared in the past decade.

The definite ancestor for computer text entry methods is a typewriter. The idea of the typewriter was first mentioned in the patent from 1714 by Henry Mill who defined a "*Machine for Transcribing Letters*" as "*an artificial machine or method for the impressing or transcribing of letters singly or progressively one after another*" [141]. There is, however, no evidence that such machine ever existed.

The first typewriter was constructed by Pellegrino Turri in 1808. Not only it was the first typewriter, it was actually invented as an assistive technology for a blind countess. The first commercially successful typewriter was constructed much later by Malling Hansen in 1870.

The most influential typewriter was constructed in 1874 by Christopher Latham Sholes and Carlos Glidden. Their QWERTY layout of characters [172] is still used with little modifications on computer keyboards today, even though there were many attempts to improve the layout (e.g., Dvorak's layout [29]).

With the emergence of the first computers, the typewriter became a computer input device. Use of such device is first documented with the BINAC computer in 1948 [193]. Later, the typewriter as input device turned into a terminal keyboard (e.g., Multics computer system, 1964). Today, keyboard is an essential input device of personal computers and laptops.

A common feature of all keyboards described above is the fact that they have been operated by physical key presses. This is not necessarily the only way to enter text. Number of different input modalities exists, which have already been used for this purpose. Although QWERTY keyboard is still dominant in desktop computing, other input modalities appeared to be more appropriate in some special cases such as mobile environment or assistive technology. For example, expansion of pointing devices (mouse, touchpad, head tracking, eye tracking) led to design of many on-screen keyboards.

One of relatively novel interaction modality is the acoustic input. The most effort within the scope of acoustic input has been given to speech recognition. First simple speech recognizers appeared in 1950s and 1960s. They were capable of classifying only several sounds (vowels, consonants) or several words [78]. Thanks to advances in pattern recognition and statistical modeling, speech recognizers have become more and more reliable and usable.

Currently, speech recognition is used in many areas including automotive industry, telephony, mobile environments, and computer accessibility. However, people with speech impairments still experience quite low accuracy when using the speech recognition software. For them, *non-verbal vocal input* (NVVI) can be a reasonable choice. In the NVVI, the user interacts with a computer application by sounds other than speech, such as humming, hissing, or blowing. Various features of the input sound are extracted and used as commands in the interaction.

1.1 Motivation

Human society is becoming more and more dependent on computers. We use them at work, in our free time, we use them for shopping, reading news, managing bank accounts, communicating with other people, etc. Using computers is not difficult for most people, however, there is a significant group of people with disabilities for whom the use of computer might be difficult or even impossible. Those people can benefit from an assistive technology. For example, blind people can use screen readers and braille displays, people with motor impairments can operate computers by speech commands.

The need of socialization belongs to one of fundamental human needs. However, for severely impaired people it might be difficult to satisfy this need as their ability to communicate could be limited. For such people, assistive technology might play an inevitable role in mediating communication with other people.

According to *Word Health Organization*[1] approximately 10% of the world's population experience some form of disability or impairment. There is a whole range of disabilities—visual impairments, motor and dexterity impairments, hearing impairments, cognitive and mental impairments, etc. The seriousness of impairments may also vary from mild to serious ones. People with impairments are often excluded from use of certain objects including ICT devices. An interesting tool called *Exclusion Calculator* [208] is capable of computing number of such excluded people in Great Britain by specifying their capabilities and degree of their impairment.

Physical impairments [50] are caused mainly by traumatic injuries, diseases, and congenital conditions. Spinal cord injuries can cause malfunction of legs (paraplegia) or all limbs (quadriplegia). Loss or damage of upper limbs is another traumatic injury that affects work with computers. People with cerebral palsy can experience spans, involuntary movement, impaired speech and even paralysis. Muscular dystrophy is a disease, in which muscles

[1] http://www.who.int

are progressively degenerated and also can lead to paralysis. People with multiple sclerosis experience different sets of symptoms such as tremors, spasticity, or muscle stiffness. Spina bifida also causes motor difficulties that can lead to paralysis. Amyotrophic lateral sclerosis causes slowness in either movement or speech. The elderly can be often handicapped by arthritis. Pain in joints affects fine motor control. Parkinson's disease and essential tremor cause uncontrollable tremors and affect voice in more severe cases as well.

In case of severe physical impairment, people usually have to use another interaction modality to substitute traditional input devices. The term *modality* (or interaction modality) refers to a path of communication between the human and the computer, e.g. speech recognition used for input. Nigay [135] defines modality as a couple of a physical device and an interaction language. A device either acquires or delivers data and an interaction language defines a set of conventional assembly of symbols that convey meaning. For example, a graphical input modality is described as the couple *(mouse, direct manipulation)*. Typically people interact with a computer using more than one modality (e.g., typing on a keyboard, pointing with a mouse). Such combination of two or more input modalities is then called *multimodal* interaction [143].

One of modalities that can be used by people with specific degree of motor impairment is the acoustic modality [182], namely the acoustic input. The acoustic input includes particularly two input modalities: automatic speech recognition (ASR) and non-verbal vocal input (NVVI).

Motor-impaired people can benefit from the speech recognition. The typical ASR software for motor-impaired users enables access to mouse, keyboard, and operating system shortcuts by uttering simple speech commands which contain one or two words [138]. Despite attempts to provide systems for natural language dictation, motor-impaired people still rely on simple speech commands as they are simpler to recognize and the recognition accuracy is high.

In the non-verbal vocal input (NVVI), the user controls computer by other sounds such as humming, hissing, or blowing. Specific features of a sound such as tone length, timbre, volume, or pitch are extracted and used as different commands for the interaction. The NVVI can be used either continuously with a real-time feedback or discretely when specific voice patterns are translated to commands similarly to ASR.

The acoustic input can be used either as a standalone modality or multimodally in combination with other modalities. This holds for both, ASR and NVVI. The first multimodal system, which used ASR and pointing, is called "*Put that there*" [18]. Multimodal interfaces which used speech as one of modalities has gained attention from the research community in the past three decades in accessibility [96, 160, 49] as well as other areas [144]. Only several works exist on using NVVI in multimodal interaction so far—a mouse emulation [A5], stylus [60] and touch [162] augmentation.

1.2 Challenges

The main goal of this thesis is to study how NVVI can be applied to novel text entry methods and techniques in order to improve quality of life of motor-impaired people. This goal involves several aspects that need to be explored in order to build a full picture of the problem. Solving each aspect brings us towards better understanding of the technology, target users, and applicability of NVVI and text entry methods in particular contexts.

Aspect 1: Combining text input and NVVI. When focusing on text input, the main drawback of NVVI is that the number of distinct NVVI commands is much lower than number of words in a dictionary of an ASR system. Therefore, we can hardly assign a simple sound to each letter and more sophisticated text entry methods have to be used. A text entry method has to be properly designed and optimized for the non-verbal vocal input.

Aspect 2: Text input optimization. Design of a text entry method has many parameters that need to be studied, for example, use of prediction, layout design, number of NVVI input patterns, letter arrangements, etc. Each parameter influences to a certain extent the entry rate, error rate, and a subjective rating. We may thus not only ask which method is optimal, but also which combination of the parameters of the method yields the best performance for particular target group.

Aspect 3: Applicability of NVVI. It is widely believed that NVVI is suitable for motor-impaired people (e.g., [14, 185]), however, the suitability for this target group has not been studied much. Motor-impaired people use speech recognition software for interacting with computers. The NVVI can be then used as a complement to speech recognition for real-time tasks (e.g., pointer movements). However, every person with motor-impairment is different with different needs and abilities. We may still ask who benefits most from the NVVI? What is an "optimal target group" of NVVI and where is the border of that target group?

Aspect 4: Acceptability of NVVI. NVVI interaction patterns were designed more or less in an ad hoc way so far. We may then ask if some of the patterns are more acceptable and suitable for particular users. We can compare perceived fatigue, satisfaction, and efficiency when using these patterns. The acceptability may be studied in multiple contexts and environments or for various target groups.

Aspect 5: Accuracy of NVVI. Previous research has shown that people are very sensitive on accuracy of an interactive system. Poor accuracy may negatively influence the acceptability as well. The NVVI should be thus highly accurate. The accuracy of NVVI is

affected by human-oriented and technology-oriented aspects. When focusing on the human aspect, the accuracy may be improved by designing optimal NVVI interaction patterns which are easy to learn and simple to produce. When focusing on the technology, the accuracy may be improved by development of robust methods of audio signal processing.

1.3 Contributions of the Thesis

The thesis contributes to the field of accessibility by combining non-verbal vocal input and text entry methods for motor-impaired people. As shown above, this problem has multiple aspects to be solved. These aspects are covered by contributions of this thesis as follows:

1. *Predictive keyboard.* Several novel text entry methods for predictive text input using NVVI are presented and evaluated with disabled participants [A4, A8]. *Aspects 1–4* are covered by this contribution.

2. *Ambiguous keyboard.* A study of a NVVI-operated ambiguous keyboard with disabled participants is presented [A6]. *Aspects 1* and *3* are covered by this study.

3. *Scanning keyboard.* Two novel text entry methods for NVVI are presented: N-ary scanning, and row-column scanning on an array [A7]. These two methods retain static character layout even though contextual character probability is used. *Aspects 1* and *2* are covered by this contribution.

4. *Comparison of NVVI commands.* A comparison of NVVI commands from the subjective point of view of the users (published in [A1]). General guidelines for design of NVVI systems were built based on this comparison for able-bodied people. The study covers *aspects 4* and *5*.

5. *Segmentation of speech and humming.* A novel method for real-time segmentation of speech and NVVI (published in [A2]). The method is capable to filter out spontaneous speech in NVVI or it can be used for future application combining speech and NVVI patterns. The method covers *aspect 5* as listed in the previous section.

1.4 Dissertation Organization

The thesis is organized into three parts: state of the art, contributions, and appendices. The state of the art part contains two chapters aiming on the overview of non-verbal vocal input (Chapter 2) and on the overview of text input for motor-impaired people (Chapter 3).

Chapters 5–9 describe the contribution of this thesis. Chapter 5 presents a novel method for segmentation of speech and humming signal. Chapter 6 describes a study of NVVI commands and presents design guidelines for NVVI. These results are applied in the next

three chapters which describe three text entry methods operated by NVVI. Novel scanning keyboards are introduced in Chapter 7, an ambiguous keyboard is studied in Chapter 8, and a novel predictive keyboard is described in Chapter 9. Conclusions and future work are described in Chapter 10.

Appendix A presents a formal description of NVVI, which has been used in the contributions to match the correct NVVI commands. Appendix B is a list of abbreviations and acronyms used in the thesis.

Part I

State of the Art

2 Non-Verbal Vocal Input

As mentioned in the previous chapter, the thesis connects two research fields: the non-verbal vocal input and text entry methods, both in the area of accessibility. Therefore, the state of the art is divided into two chapters describing each field separately. This chapter is dedicated to a general description of non-verbal vocal input and provides an overview of all NVVI applications found in the related literature. The text entry methods with special focus on motor-impaired people are described in the next chapter.

The *non-verbal vocal input* (NVVI) can be described as an input modality, in which the user interacts with a computer application by sounds other than speech. The interaction depends on specific features of the sound, for example, pitch of a tone, length of a tone, volume, or timbre. The NVVI has already received a significant focus within the research community. It shares some similarities with speech input (performed by automatic speech recognition, ASR). It utilizes vocal tract of the user and a microphone that picks the audio signal. However, both interaction modalities are better fitted to different scenarios, therefore NVVI should be considered as a complement to speech input rather than its replacement. When comparing NVVI and speech input, several differences can be found:

- NVVI is better fitted to continuous control rather than speech input. There has been several studies published that support this statement [184, 59, 55].
- NVVI is cross-cultural and language independent [188].
- NVVI generally employs simple signal processing methods [72].
- NVVI has limited expressive capabilities, speech input is better at triggering commands, macros or shortcuts [181].

In this thesis, the term *vocal gesture* refers to a single indivisible unit of interaction within NVVI. A vocal gesture is then interpreted in the target application as a single command. Vocal gestures can be either simple or complex. Simple vocal gesture is a continuous sound produced by the user delimited by silence. A complex vocal gesture is composed of two or more simple vocal gestures. The difference is shown in Figure 2.1.

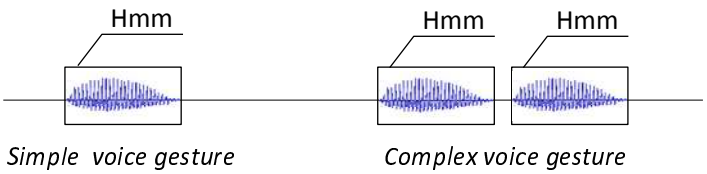


Figure 2.1: A simple vocal gesture followed by a complex vocal gesture.

2.1 Classification of Non-Verbal Vocal Input

The non-verbal vocal input can be classified from two points of view: features of the sound signal, and type of input channel. The former point of view focuses on which feature of the signal is used to trigger an action (pitch, volume, timbre, length). The latter point of view focuses on how the feature is used (either continuously or as events).

2.1.1 Sound Signal Features

The interaction within an application operated by NVVI may depend on four basic features of the sound: pitch, timbre, volume, and length. Each feature may be used independently or they can be used in a combination. Combination of length and the other three features is commonly used. Combination of pitch, timbre, and volume have been used rarely.

Pitch. Pitch expresses the height of a tone and is measured by fundamental frequency of a sound signal. Pitch can be extracted from sounds like humming, whistling, or singing [185].

Timbre. Timbre is a feature which allows us to recognize among sounds from different sources (e.g., piano sound vs. violin). The timbre is usually extracted from a frequency spectrum of the input signal. In NVVI, the timbre-based input usually differentiates among vowels (e.g., "*aaa*" or "*ooo*") or consonants (e.g., "*ck*" or "*sss*") [14].

Volume. Volume refers to loudness of the input audio signal [149]. It can be roughly computed as energy of the signal.

Length. Length is a period of time, for which the sound was produced by the user [1].

2.1.2 Input Channel Types

The applications of the non-verbal vocal input can be roughly divided into two categories: real-time and non-real-time. The real-time applications (**continuous input channel**) allow the user to receive immediate feedback while still producing the sound, which is useful, for example, in computer games [184, 62, 55], interactive art installations [3], or mouse emulation [185, 14]. NVVI thus works in contrast to the speech recognition where the system waits for completion of an utterance.

In the non-realtime applications (**event input channel**) of NVVI, users are expected to finish producing the non-speech sounds before the system responds. The interaction with these systems follows the query–response paradigm, similar to the speech-based systems. These sets of applications are important for people who are not capable of sufficient level

of speech articulation required by the current automatic speech recognizers. Moreover, in event input channel several simple vocal gestures can be combined to a complex vocal gesture in order to extend the number of distinctive commands (e.g., [181]).

Continuous Input Channel. Igarashi and Hughes [72] proposed the use of non-speech sounds to extend the interaction using automatic speech recognition (ASR). They reported that non-speech sounds were useful to specify "analog" parameters. For example, the user could produce an utterance such as "volume up, *aaah*", to which the system would respond by increasing a volume level as long as the sound would be held. Similar approach could be used with speech-operated computer mouse, for example, "move left, *hmmmmmmm*" where *hmmmmmmm* defines the number of pixels to move. Similar approach was used in work by Mihara et al. [124].

An emulation of computer mouse operated exclusively by NVVI is described by Sporka et al. [185]. This system was evaluated in a longitudinal study by Mahmud et al. [112]. Different non-verbal gestures control the movement of the mouse cursor as well as the mouse buttons. A similar approach has been used by Bilmes et al. [14].

NVVI was successfully employed as means of control of computer games [55]. Sporka et al. [184] demonstrated how the game Tetris can be controlled by humming. Al-Hashimi describe an NVVI-controlled plotter [1].

Event Input Channel. The event input channel has not been studied as extensively as the continuous input channel. It has been used, for example, in mouse emulation applications to produce the mouse clicks [185, 14]. The only work entirely based on event input channel of NVVI is the emulation of QWERTY keyboard [181] in which key was assigned to a specific complex vocal gesture or series of simple vocal gestures.

Another example of event input channel are systems for querying a database of music tracks by singing or humming a tune [46, 97, 19, 226, 153]. The tracks are indexed according to the melodies which they contain. After a melody is sung, the system searches the database and presents the user with required information.

Watts and Robinson [212] proposed a system where the sound of whistling triggers the commands in the environment of a UNIX operating system.

2.2 Pitch-Based NVVI

The thesis mostly focuses on vocal gestures operated by humming, which is a sound of "*hmmmm*" when lips are closed. Humming belongs to the pitch-based NVVI, in which the computer is controlled by the fundamental frequency of a sound signal. The pitch-based input has been used as an input modality for people with motor disabilities [14, 185, 181] as well as intonation training tool [55] for children. In these applications, vocal gestures

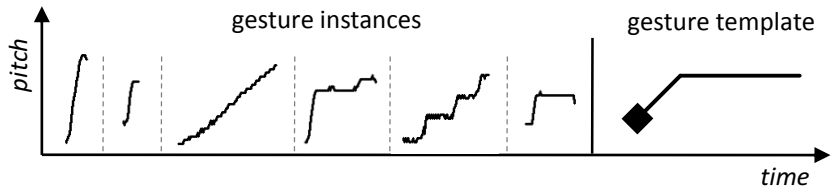


Figure 2.2: Relationship between a gesture template and its instances.

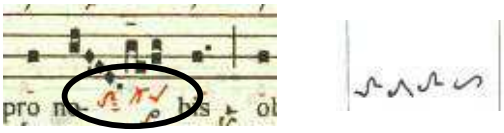


Figure 2.3: Cheironomic neumes, 9th century AD.

are defined as short melodic and/or rhythmic patterns.

When designing a set of vocal gestures, an ideal pitch profile for each gesture has to be described. These ideal pitch profiles are then referred to as *gesture templates* and they are usually represented in graphic form as shown in Figure 2.2. However, the users are unable to precisely interpret the template and produce such ideal pitch profile. Interpretations of gesture template by the user is referred to as *gesture instances*. An example of the relationship between a gesture template and its instances is depicted in Figure 2.2. Note that slightly different instances share the same meaning defined by the gesture template—all of them are rising tones.

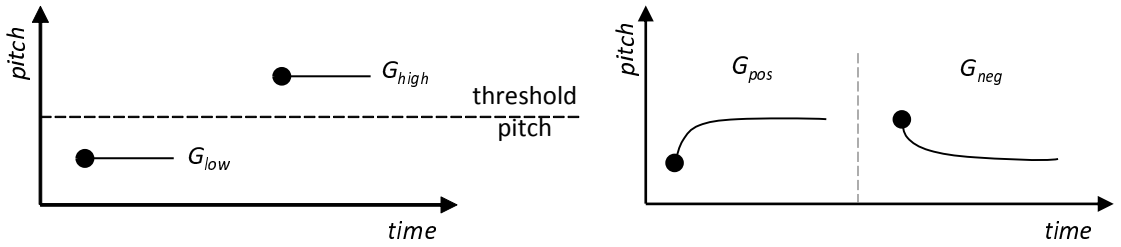
A gesture template can be described either verbally (e.g. "*make a rising tone*") or graphically (see Figure 2.2) or formally (see Appendix A). An interesting inspiration how to describe vocal gestures graphically may be found in the music notation from the 9th century [21], long before the modern notation has been adopted. The system of cheironomic neumes, used especially for the notation of spiritual chants (see an example in Figure 2.3) provided relative description of pitch and length of particular words and syllables in the chant.

This notation, though imprecise from today's view, is exactly the way how gestures are graphically represented. Because of varying voice capabilities and imprecise interpretation of the gestures the gesture description (gesture template) must cover a wider range of concrete sounds (gesture instances) produced by the users without special music education to gain an acceptable level of usability of the system.

2.2.1 Pitch-Based Vocal Gestures

The interaction that utilizes pitch-based vocal gestures can depend on an absolute pitch, relative pitch or combination of both.

In absolute pitch mapping, gestures are differentiated by the pitch of tones. Usually, this



(a) Absolute pitch (b) Relative pitch

Figure 2.4: a. Absolute pitch approach based on a threshold pitch. b. Relative pitch approach based on tonal inflections.

mapping needs a calibration to adjust to the pitch range of different users. An example of such gestures is depicted in Figure 2.4a. A single tone produced by the user is either recognized as gesture G_{high} or gesture G_{low}, depending whether the tone is produced above or below a threshold pitch. Similarly, the vocal range can be split into more subranges. However increasing number of subranges can lead to higher error rates, as more precise intonation is needed [186].

Threshold values in absolute pitch approach must be adjusted for each individual user because vocal range of the user varies. For example, difference between male and female voice is as much as one or two octaves. Moreover, intonation ability of an unskilled person is limited.

In relative pitch mapping, gestures are differentiated by relative pitch of specific components of the gesture. Figure 2.4b shows two gestures which are recognized when the pitch is rising (positive tonal inflection, G_{pos}) or falling (negative tonal inflection, G_{neg}). Relative pitch gestures do not need to be calibrated for each user as absolute pitch gestures. However, it appears that tonal inflections are for some users more difficult to produce than flat tones. This issue is discussed in detail in Chapter 6.

Combination of both approaches can be used when we need to increase number of different stimuli in the input. For example, in Whistling User Interface [185], the axis of movement is determined by absolute pitch and then the direction by relative pitch.

2.3 Overview of NVVI Applications

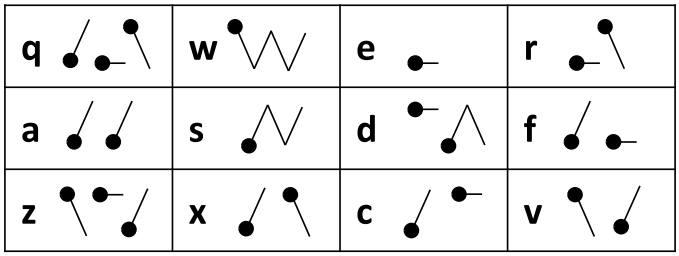
In this section, various applications of NVVI are described including keyboard emulation, mouse emulations, games, artistic installations, etc.

2.3.1 Keyboard Emulation

The keyboard emulation is a typical representative of the event input channel Sporka et al. [181] describe keyboard emulation of 39 keys including alphanumeric characters,

	group 1				group 2			
	1	2	3	4	1	2	3	4
1	q	w	e	r	t	y	u	i
2	a	s	d	f	g	h	j	k
3	z	x	c	v	b	n	m	p

(a) Pitch-to-address mapping



(b) Pattern-to-key mapping

Figure 2.5: a. An excerpt of the pitch-to-address method. A letter is typed after specifying column, group, and row by three tones of four pitches. b. An excerpt of the pattern-to-key method. A letter is typed by producing corresponding complex vocal gesture.

space, backspace, and enter. Two modes were used that differed in mapping of the vocal gestures to keys: *pitch-to-address*, and *pattern-to-key* mappings.

In the pitch-to-address method (see Figure 2.5a) a sequence of hummed tones (simple vocal gestures) is considered as a vector of coordinates of keys in a keyboard layout. The comfortable pitch range of each user is split into four subranges. A key can be than accessed by producing sequence of three tones. The first tone is used to select a row of keys, the second tone determines group of keys and the last one specifies column. This methods enables addressing of $4^3 = 64$ different keys.

The pattern-to-key mapping method assigns a unique complex vocal gesture to each key. Each complex vocal gesture consists of set of simple vocal gestures. Two sets are used:

- Morse alphabet. Only simple tones differentiated by length were used: short tones (dots), and long tones (dashes).
- Artificial primitives (see Figure 2.5b). The vocal gestures are composed of simple primitives such as low and high flat tones and raising and falling tones. The most frequent keys were accessible by the simplest gestures, as frequency distribution of the letters in English was kept in mind.

The keyboard emulation was verified in a user study, in which 9 people without disabilities took part. Morse alphabet mapping was the fastest and perceived as the best, however, participants made less errors when using pitch-to-address mapping. Pattern-to-key mapping was the slowest and subjectively perceived as the worst [181].

2.3.2 Pitch-Based Mouse Emulation

Sporka et al. [185] designed a mouse cursor emulation called *Whistling Mouse*, which is controlled by whistling. Two different modes are defined: orthogonal and melodic, each

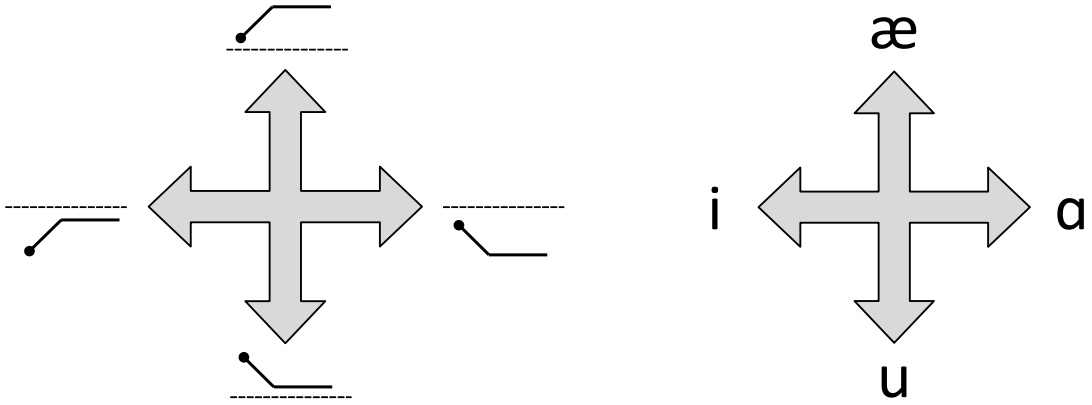
mode defined different gesture sets for the cursor movement. Short tone for emulating mouse click is used in both modes.

In the orthogonal mode (see Figure 2.6a), the mouse pointer is moved either horizontally or vertically. The axis of movement is determined by the initial pitch of the tone. If a tone is started below a specified threshold pitch, the mouse pointer can be moved along the horizontal axis. Similarly, if a tone is started above the threshold pitch, the movement is limited to vertical axis. The direction of the pointer is determined by tonal inflection—rising tone makes the pointer move up or to the right according to absolute pitch of the initial pitch. Falling tone makes the pointer move down or to the left. The speed of the mouse pointer is directly dependent on the magnitude of difference of the initial and actual pitch.

The melodic control mode allows the cursor to move in any direction according to the pitch with constant speed. Change in the actual pitch affects the azimuth in which is the cursor moving. Producing the base tone, which should be approximately in the middle of the user's vocal range [183], moves the cursor up and higher or lower tones make the cursor move right or left respectively.

Another mouse emulation was presented by the author of this thesis [A5]. The emulation is a multimodal system in which head tracking and NVVI are used. The head tracking controls position of the mouse pointer while NVVI is used for mouse clicks. The NVVI is capable of simulating almost all actions which can be done with the mouse: left click, right click, double click, as well as more complicated actions such as drag and drop operations, and scrolling.

Chanjaradwichai et al. [22] investigated in a mouse emulation based on mouse grid [25] and compared humming vocal gestures to speech commands. They found that humming



(a) Whistling Mouse (b) Vocal Joystick

Figure 2.6: a. Orthogonal mode of the Whistling Mouse. Initial pitch determines the axis and tonal inflection determines the direction. b. Vocal Joystick. The direction of the movement is determined by vowel produced by the user.

performed better probably due to worse accuracy of speech recognition. This approach was modified in their following work [23] to scanning menus which allowed to control some basic tasks in the MS Windows operating system.

2.3.3 Timbre-Based Mouse Emulation

Timbre-based input is another part of the NVVI, in which the computer is controlled by quality of sound recorded by microphone. Timbre is a feature of a sound signal that allows to distinguish among various sounds, for example, sound of different musical instruments.

Several papers have been published reporting on emulation of mouse by producing vowels. The method is called *Vocal Joystick* [14] and is shown in Figure 2.6b. The quality of vowel is classified and mapped to one of eight directions. For example, vowel "*u*" is mapped to movement down, "*æ*" is mapped to movement up, "*a*" to upper left etc. Volume is also extracted from the audio signal and it controls the speed of mouse cursor—the softer the sound, the slower the cursor movement and vice versa. The cursor movement of Vocal Joystick can be modeled by Fitts' law [101]. The study showed that expert index of performance is approximately a third of a computer mouse. The performance is comparable to standard joystick. A longitudinal use of the Vocal Joystick was studied by Harada et al. [63]. The performance of Vocal Joystick is comparable to that of Whistling Mouse [185] as found in another longitudinal study conducted by Mahmud et al. [112].

2.3.4 Gaming

People with motor disabilities are disadvantaged in playing arcade games as a rapid reaction is often required. Speech recognition can be suitable for those people, however, the response delay makes this technology unusable for playing real-time games, as the recognizer has to wait for the user to finish their utterance. Real-time games are usually operated by limited number of commands (e.g. movement keys and action key), therefore vocal gestures can be easily mapped onto these commands.

One example of game played by pitch-based input is the *Hedgehog game* [55] for children. In this interactive game, children are encouraged to learn singing. They have to sing along music played and move an avatar. The current pitch is aligned with vertical position of the avatar. Another game presented is the pitch-controlled *Pong* [55], in which pitch is aligned with vertically moving bat. The goal is to intercept and bounce a ball.

The advantage of NVVI over speech input was demonstrated in work by Sporka et al. [184]. *Tetris* game[1] was used to compare both modalities. The results showed that the NVVI control was about 2.5 faster and up to three time more accurate than speech. The study indicated that NVVI is more suitable for the control of real-time games with limited number of commands. A similar study which compared speech and NVVI control was done

[1] "At 25, Tetris still eyeing growth". Reuters. June 2, 2009

by Sporka and Slavík [187]. In this study, a radio-controlled car model was operated by the participants. The results were similar to the Tetris study [184].

Harada et al. [62] compared a vowel and speech control of three games. They found similarly to Sporka et al. [184] that vowel control is better suited for real-time control than speech.

2.3.5 Mobile applications

Won et al. [221] described a subvocal humming system for mobile phones capable of operating specific commands such as dialing a number, answering a phone call, playing music, etc. The subvocal signal pickup is unobtrusive and immune to external environmental noise which makes it optimal for mobile applications.

Voice augmented manipulation [162] is a multimodal technique for mobile phones. This technique incorporates touch and NVVI vocal gestures in order to enhance certain commands on mobile devices (zooming, panning, scrolling, ...).

2.3.6 Other NVVI Applications

VoicePen [60] is an application for drawing, in which the input of a digital pen is augmented by timbre-based NVVI. Creative drawing task can be performed by the digital pen, while the kind of vowel can control opacity or brush thickness. The NVVI can also control object rotation, translation, and zooming.

VoiceDraw [61], another application for drawing, is controlled entirely by the timbre-based NVVI. The drawing cursor is controlled by the quality of vowel controls similarly to the Vocal Joystick [14]. Volume of the sound can be used to control one of the stroke attributes, such as thickness or color.

Robotic arm in a simulated environment [116] is controlled by the quality of a vowel, pitch and volume. The arm has three joints, which can be rotated in one dimension (clockwise or counterclockwise). Two of them are controlled by vowels and one of them by pitch. Rotation speed is determined by the volume of the sound. Control of a real 3D robotic arm in a similar way is described by House et al. [69].

Al-Hashimi [3] presented several applications that transform voice to physical actions. They are based only on the length of the sound. *SssSnake* is played on a table by two player. Virtual snake is projected onto the table. The direction of snake movement is controlled by uttering "*sss*" into one of four microphones that are mounted on edges of the table.

Expressmas Tree [2] is a real Christmas tree, in which real light bulbs can be switched by producing a continuous sound (e.g. blowing, hissing). A simple game was designed for this installation. The goal of the game is to switch on a given number of bulbs. The research also reports on subjective perception of sounds of different timbre (e.g., "*hmmm*",

NVVI	Able-bodied	Motor-impaired	Speech and motor-impaired
Humming	12	-	-
Vowels	8	2	-
Hissing	1	-	-
Blowing	1	1	1

Table 2.1: Summary of types of NVVI input and number of studies with different target groups.

"*aaah*", "*sss*") by shy and outgoing persons. *Blowtter* [1] is a voice-controlled plotter. Four microphones are used to control movement of the plotter. Speech commands are used to raise the pen and to move the pen into the paper. The *Blowtter* was successfully used by motor-impaired children.

Perera et al. [149, 150] presented another drawing application for disabled artists controlled exclusively by the volume level of the produced sound. The drawing is done in similar manner to the melodic control of the Whistling mouse [185].

2.4 Summary

As shown in this chapter, the NVVI has been studied and accepted as an input method. However, an important question is what is the ideal target group of NVVI users? Based on empirical evidence, we can see that the acceptance of NVVI is low as the users are embarrassed to produce NVVI sounds, especially in public.

Therefore, most of the previous research point out that NVVI is suitable for motor-impaired people with upper limbs impairments. However, these people can use automatic speech recognition (ASR) systems and NVVI can be used only as a complement of these ASR system. ASR systems show usually poor accuracy for people with speech impairments. A question then arises whether the NVVI would be suitable for people with combined motor and speech impairment.

Table 2.1 shows that even though researchers claim that NVVI is suitable for motor-impaired people, only several studies has been actually conducted, which have included participants form this target group. Only vowels and blowing was successfully tested with motor-impaired people. Blowing was tested with one speech-impaired participant only. This thesis focuses on humming and hissing and evaluates whether these two types of NVVI are suitable for motor-impaired people and people with combined motor and speech impairment.

3 Text Input for Motor-Impaired People

Previous chapter describes the state of the art in the area of non-verbal vocal input (NVVI). However, this thesis focuses not only on the NVVI. The NVVI is taken as an interaction modality in the specific use case—text input for motor-impaired people. This chapter describes the second part of the state of the art of the thesis as it summarizes various text entry methods with special focus on motor-impaired people.

This chapter provides an overview of 150 publications on text input for motor-impaired people and describes current state of the art. We focus on common techniques of text entry including selection of keys, approaches to characters layouts, use of language models, and interaction modalities. These aspects of text entry methods are further analyzed and examples are given. The chapter also focuses on an overview of reported evaluations by describing experiments, which can be conducted to assess the performance of a text entry method. After that, we give a summary of 61 text entry methods for motor-impaired people found in the related literature and classify them according to the aforementioned aspects and reported evaluation types. We show that setup of text entry experiments varies in many publications and that the reporting of results (type rate, error rate) is also not standardized. This makes the methods difficult to compare. Thus, we express a need for a unified text entry experiment which would standardize the procedure and metrics.

3.1 Introduction

Text input is a common activity of many ICT devices such as laptops, phones, or tablets. Even though most people find entering text easy and natural, it is challenging for people with certain disabilities. In order to support inclusion of people with disabilities into society, it is important to offer text entry methods, which are appropriate for their needs. This chapter gives an overview of existing methods and techniques covering a broad range of groups of motor-disabled people.

Many different physical impairments exist causing different disabilities from deteriorated finger dexterity to complete paralysis. Physical impairments are consequences of traumatic injuries, diseases, and congenital conditions. Spinal cord injuries can cause malfunction of legs (paraplegia) or all limbs (quadriplegia). Loss or damage of upper limbs is another traumatic injury that affects work with computers. People with cerebral palsy can experience spasms, involuntary movement, impaired speech and even paralysis. Muscular dystrophy is a disease in which the muscles are progressively degenerated and also can lead to paralysis.

People with multiple sclerosis experience different symptoms such as tremors, spasticity, or muscle stiffness. Spina bifida causes motor difficulties that can lead to paralysis. Amyotrophic lateral sclerosis causes slowness in either movement or speech. Elderly people can be often handicapped by arthritis. Pain in joints affect fine motor control. Parkinson's dis-

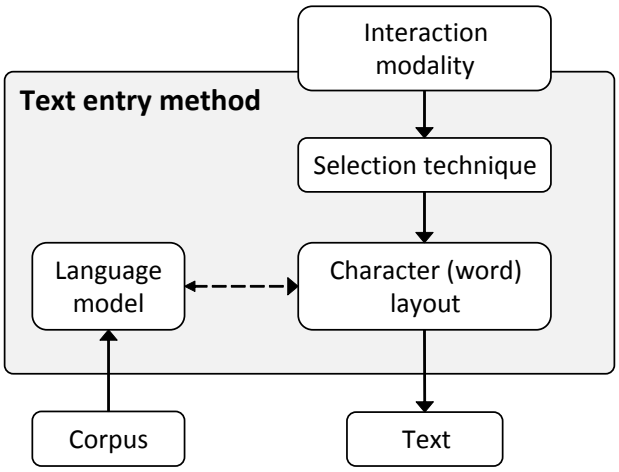


Figure 3.1: A simple model of a text entry method

ease and essential tremor cause uncontrollable tremors and affect the voice in more severe cases as well.

Wide range of text entry methods appeared in recent years. To help such diverse group of people, many techniques and interaction modalities are used in these methods. Thus, we may not determine one dominant text entry method for motor-impaired people. Figure 3.1 shows common features of a typical text entry method. The essential function of every text entry method is to provide a mean for inputting text by selecting individual characters or words. Common techniques for character and word selection are described in Section 3.2. A character or word is selected from a method-specific layout or distribution which is discussed in Section 3.3. The layout of characters or words usually depends on a language model. The use of language models is described in Section 3.4. In order to make the character selection possible, an appropriate interaction modality has to be used. Interaction modalities suitable for motor-impaired people are summarized in Section 3.5.

Text entry methods are usually subject to experiments and evaluations. Possible experimental setups and typical evaluations are discussed in Section 3.6. In Section 3.7 we summarize 61 methods found in the related literature. The methods are classified according to the type rate, selection technique, character layout, language model, and interaction modality. Discussion regarding the experiment setups of these methods is also presented in this section.

3.1.1 Related Overviews

Several overview papers exist focusing on text entry in augmentative and alternate communication. For example, Boissiere and Dours [16] published in 2003 an overview of existing writing assistance systems focusing mostly on prediction and language modeling. A short overview with a special focus on conversation modeling was published by Arnott [7]. Trewin

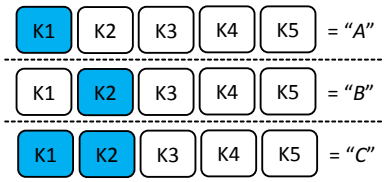


Figure 3.2: Chording keyboard with five keys. The blue background indicates pressed keys.

and Arnott [197] describe mostly specialized keyboard hardware and on-screen keyboards for motor-impaired people. Other overviews focus on ambiguous keyboards for augmentative and alternate communication [79], scanning systems [103, 140], and eye typing [115].

The overviews listed above describe only a subset of related literature focusing on a particular technique or modality. The overview presented in this chapter, on the other hand, gives an exhaustive summary of techniques and methods for motor-impaired people. Moreover, we describe and review evaluations reported in the literature.

3.2 Selection Techniques

On the first typewriters, typing a character was simply done by pressing a key. When *shift* was added, the number of characters to enter became greater than the number of keys the typewriter provided. Recent mobile phones with 12 keys allowed to type almost the same amount of characters as a standard PC keyboard with approximately 100 keys.

As mentioned above, number of different impairments with rather diverse consequences exist. While some people are completely excluded from using the PC keyboard, number of people can still use it despite achieving slow type rates. For these people, reducing number of keys might be one way of increasing type rate.

Every text entry method has some kind of selection technique to determine the character to be entered. In the literature, we identified the main selection techniques as follows: *direct selection*, *scanning*, *pointing*, and *gestures*.

3.2.1 Direct Selection

Direct selection is a technique, which enables the user to select directly a key out of a set of keys. The set of keys available for motor-impaired people is usually limited. When reducing the set of keys, three basic techniques can be used: chording keyboard, ambiguous keyboards, and encoding.

Chording keyboards reduce the number of keys but require the ability to press multiple keys at the same time. Different combinations of pressed keys correspond to different characters typed which results in relatively high type rates [98]. This selection technique, however, is rarely used for motor-impaired people as reduced dexterity of their hands hinders them

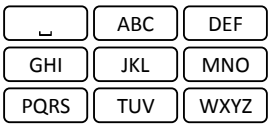


Figure 3.3: Ambiguous keyboard. Multiple letters are assigned to one key.

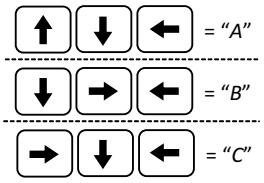


Figure 3.4: An example of encoding. Four arrow keys (left, right, up, down) are used to encode letters in this example.

from accurately pressing multiple keys at the same time. A number of chording keyboards exists with number of key-to-character mappings. An example of a possible mapping is depicted in Figure 3.2. Letters "*a*" and "*b*" are entered by single keys (K1 and K2 respectively), but the letter "*c*" is entered by pressing K1 and K2 simultaneously.

Ambiguous keyboards, such as T9 [52] or Multitap [147, 105], are very popular among motor-impaired people. In these keyboards, the alphabet is divided into several groups of characters and each group is assigned to one key (see Figure 3.3). Ambiguous keyboards are described in detail in Section 3.4.3.

Encoding [77, 16] is another technique which aims at reducing number of keys. Each character corresponds to a unique sequence of key presses. The sequence is usually referred to as "code". Examples for this case are binary spelling interfaces using Morse code [190] and Huffman code [196], where only two keys are used. Other examples are MDITIM [74] or UDRL [30], both using four direction keys. An example of encoding is shown in Figure 3.4 where letters are assigned to a unique sequence of four arrow keys.

3.2.2 Scanning

When only a very low number (one or two) of keys is available, *scanning* can be used in a text entry method. Scanning systems and keyboards have been studied extensively in past decades. Scanning refers to an item selection technique in which a number of items are highlighted sequentially until the desired item is selected and the corresponding command is executed (e.g., a letter is typed). Scanning is based on two atomic operations: *scan step* and *scan selection*. The step operation highlights items one by one in a predefined order while the selection operation executes a command assigned to the highlighted item.

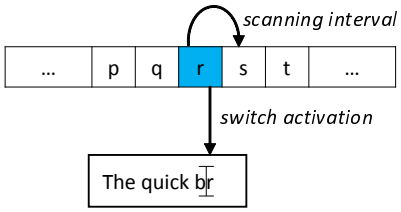


Figure 3.5: Automatic scanning. Selection is done by switch activation and step by a scanning interval.

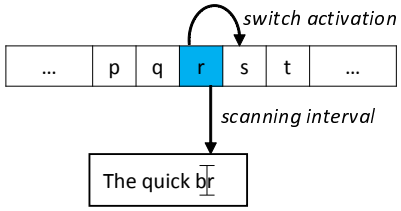


Figure 3.6: Step scanning. Selection is done by a scanning interval and step by switch activation.

We can categorize the scanning text entry methods according to two aspects: modes and techniques. A *scanning mode* determines mapping of user input on scan step and selections [127]. A *scanning technique* describes how items are grouped and how the scanning proceeds among the items.

Scanning Modes

In the simplest case, scanning requires only one unique signal from the user (further referred to as switch) which is mapped either to step or to selection. The other operation is then triggered after a predefined scanning interval is reached. Based on the mapping, we can distinguish among several scanning modes. The most prevalent are:

Automatic scanning. The automatic scanning is the most common mode. The selection is controlled by the user input (switch activation) and step is triggered automatically after a scanning interval expires. An example is shown in Figure 3.5.

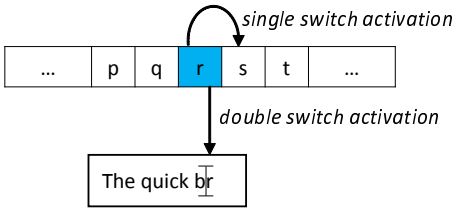


Figure 3.7: Self-paced scanning. Selection is done by a double switch activation and step by single switch activation.

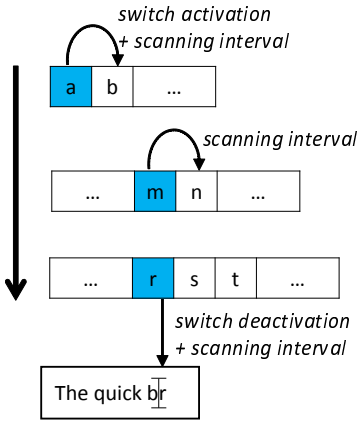


Figure 3.8: Inverse scanning. Scanning starts after switch activation (e.g., pressing a button switch). Item is selected when the switch is deactivated (e.g., releasing a button switch).

Step scanning. The step scanning [127] (see Figure 3.6) is similar to the automatic scanning, but the control of selections and steps is reversed: steps are controlled by the user and selections are automatic.

Self-paced scanning. The self-paced scanning [41] (see Figure 3.7) distinguishes from single and double switch activations. Double switch activations correspond to two consecutive switch activations issued within a short timeout. Then, the single switch activation is used as a scan step and double switch activation as a scan selection.

Inverse scanning. The inverse scanning [127, 128] (see Figure 3.8) requires a switch with two states (e.g., button is pressed or released). When the switch is activated (e.g., a button is pressed), automatic scanning is started. Once deactivated (e.g., a button is released), the selection is done by waiting for the scanning interval.

Scanning Techniques

Several scanning techniques exist, which are used not only for text entry but also for menu selections or browsing the contents of a menu.

Linear scanning (e.g. [209]) is probably the simplest technique. Items are sequentially highlighted in one group until the desired item is selected. An example of typing letter "*c*" is depicted in Figure 3.9. When two switches are available, the scanning interval is

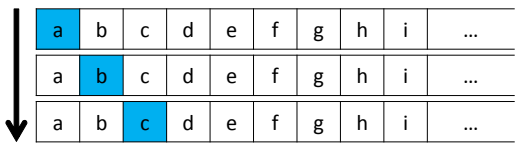


Figure 3.9: Linear scanning. Letters are highlighted sequentially in each scanning step.

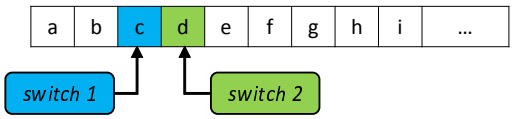


Figure 3.10: Two switch scanning. Two letters are highlighted in one scan step.

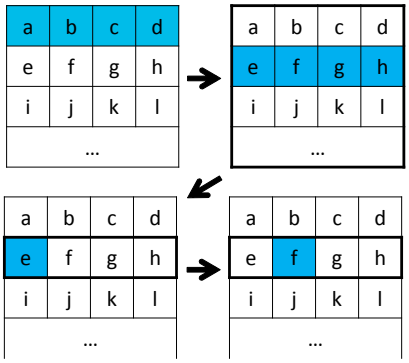


Figure 3.11: Row-column scanning. An example of selecting the letter "*f*" by one row and one column step.

usually replaced by the second switch. Another approach, which uses two switches, keeps the scanning interval and offers the user the possibility to select the current highlighted n^{th} item or the next $(n+1)^{\text{th}}$ item [157]. In example depicted in Figure 3.10, either "*c*" or "*d*" can be entered within one scanning interval by switch 1 or switch 2 respectively.

Linear scanning is very slow, especially for items at the end of the sequence. To address this issue, *row-column scanning* (e.g. [83, 164]) can be employed. In this technique, the items are organized in a matrix and selecting an item is done in two levels. In the first level, rows are sequentially highlighted until the selection is made and then items in the selected row are linearly scanned. The process of selecting letter "*f*" is depicted in Figure 3.11.

Three-dimensional scanning (or *group scanning*) [39, 95] reduces the number of scanning steps by adding one more level. In this level, groups of characters (or quadrants, see Figure 3.12) are sequentially highlighted until the selection is made. Then, standard row-column scanning is employed.

Binary scanning (or *dual scanning*) [64, 41] recursively splits items into two halves until a single item is highlighted. This technique is similar to binary search well-known from basic programming algorithms. *N-ary* scanning (see Chapter 7) is the generalization of the binary scanning. Ternary scanning was found to be optimal among other *N-ary* scanning techniques for the use case of character input. An example of typing "*n*" by ternary scanning is depicted in Figure 3.13. In the first level, the alphabet is split into three groups "*a–h*", "*i–q*", and "*r–z*". The scanning proceeds to the second group where the selection is made. In the second level, "*n*" is selected within the second scan step.

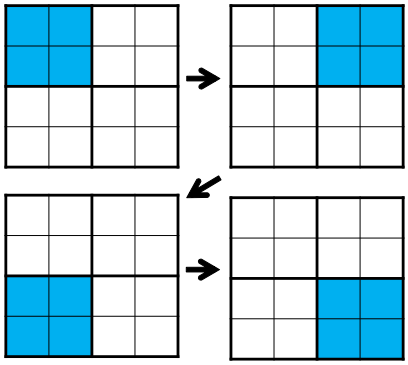


Figure 3.12: Three dimensional scanning.

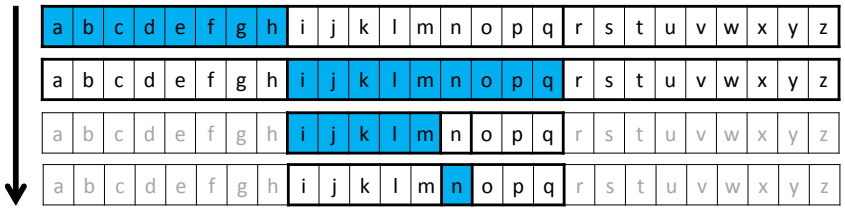


Figure 3.13: Ternary scanning.

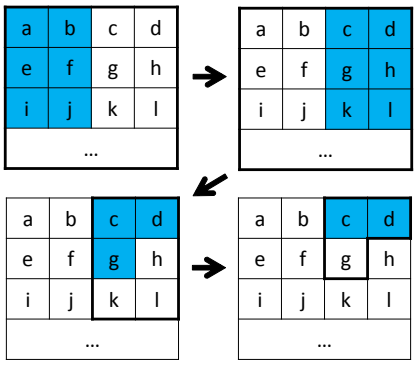


Figure 3.14: Binary Huffman tree scanning on a matrix.

Containment hierarchical scanning [9] is a model that describes any scanning system in terms of an acyclic graph. In such graph, hierarchical relations are defined—groups containing subgroups and single items. The graph defines multiple levels of scanning starting in root and proceeding down to leaves which contain characters. As soon as the leaf is reached, the corresponding character is entered. For example, the containment hierarchical model was used for scanning a binary Huffman tree [70] in work by Baljko and Tam [9] and Roark et al. [159]. An example of scanning such tree spatially organized into matrix is depicted in Figure 3.14.

The choice of scanning technique depends on the input modality, user abilities, and number of items that are scanned through. Input modality and user abilities influence the time needed to activate the switch and thus make the selection. Scanning interval has to be longer than the time required for selection. Scanning steps and scanning selections should be balanced in order to maximize the type rate. Scanning selections should be kept low as each switch activation requires the user to perform an action. Too frequent switch activation might be uncomfortable for the user.

Number of items also influences the choice of scanning technique. For low number of items, linear scanning is good enough (e.g., scanning ambiguous keyboards [104]). For an alphabet comprised of basic letters and characters, row-column scanning is usually used. When higher number of items is required, the three-dimensional scanning is a good option [39].

3.2.3 Pointing and Gestures

In some text entry methods, a pointing device controlling a mouse pointer is used to enter the text. While many motor-impaired people cannot control directly the mouse, they still can emulate it by trackballs, joysticks, head tracking, gaze, etc. Some of these modalities, however, do not support selection (e.g., head tracking, gaze interaction). Three common solutions to this problem exist: *dwell time*, *multimodal interaction*, and *gestural input*.

In the *dwell-time method*, a selection is generated after a predefined timeout when the pointer is kept within a small radius. The dwell time method is capable of simulating one selection only (usually mapped to the left mouse click). Moreover, this method raises Midas touch problem [75] which relates to the fact that the selection is always triggered when the cursor is not moving. This may lead to many involuntary selections as the user cannot rest without making one. The Midas touch problem can be solved either by adding areas where the user can stop the cursor [13] or displaying a pop-up menu after the dwell time expires [A5].

In the *multimodal interaction* approach, a different modality is used to generate the selection. Examples for this can be tracking of head features [198, 44, 200], speech recognition [96, 160, 205], non-verbal vocal input [A5], teeth clicking [225], etc. Both dwell time method and multimodal interaction can be used as a universal selection method with any pointing device.

In *gestural input*, strokes with the pointing device are usually transformed to text. Strokes can correspond to a character (e.g., EdgeWrite [218, 217, 220]), word (e.g., Quikwriting [151], SHARK [224], or Continuous EdgeWrite [120]), or even longer parts of the text (e.g. Dasher, [211]). The gestural input cannot be usually used as a universal selection method for a pointing device with an exception to gaze interaction [132].

Performance of pointing devices is modeled by Fitts' law [42], which can be used for predicting movement times based on distance to travel and width of a target. The Fitts' law is not only a robust predictor for hand movement, but also for the pointing with mouse and joystick [101], head [4], or tilt [108]. On the other hand, when applied to gaze interaction, the robustness declines due to saccadic eye movements. These movements make the movement time independent on distance [173]. In text entry systems, the Fitts' law is often used to determine the optimal layout of an on-screen keyboard operated by a pointing device, e.g. [176, 110, 223, 5, 123].

3.3 Character layouts

The ultimate goal of each text entry method is to maximize the type rate. In order to achieve this goal, researchers strive to find an optimal layout of letters or optimal code for given text entry technique, interaction style, or interaction method. A well-known example can be the Dvorak simplified keyboard [29], which is claimed to minimize finger motion in order to increase typing rate of a typewriter.

Optimal layouts are usually built by analyzing frequencies of words and letters in given corpus based on a language. Thus, a common property of optimal layouts is their language dependency. Alphabetical layouts seem to be language independent and suitable for novice users. However, the language independence of alphabetical layouts is questionable as many languages contain special characters or diacritical marks. No consensus exists regarding the suitability for novice users. A study by Norman and Fisher [136] found no improvement when comparing alphabetical to a randomized layout. Other studies [175, 47], on the other hand, confirmed that the alphabetical layout results in better performance of novice users.

The character layouts can be divided into two categories according to character distribution: static and dynamic [16]. By the term character distribution, we do not mean only visual representation of the text entry method, but rather sequence of operations, which has to be entered to type a character. These may refer to a layout of an on-screen keyboard as well as to characters encoded into a sequence of keystrokes. In static distributions, this sequence of operations remains the same while typing. In dynamic distributions, however, the sequence is gradually updated according to currently written context.

The difference between static and dynamic distributions can be demonstrated on two methods for mobile text input on a 9-button keypad. Both methods are controlled similarly as Multi-Tap method (e.g., [105, 147]) commonly available on mobile phones, in which each key is assigned to three or four characters and a character is entered by repeatedly press-

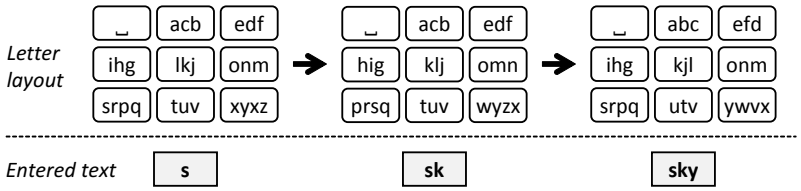


Figure 3.15: Typing "*sky*" on a keyboard with dynamic layout. Note the shifting position of letters on individual keys.

ing the same key until the desired character appears. Less-Tap [147] method uses a static layout, where the sequences of letters on the keys are sorted according to their probability in English. LetterWise [105], on the other hand, uses dynamic distribution. In this case, sequences of letters on each key are sorted according to their current probability based on already written context. The probability is updated after a letter is entered.

An example of a keyboard with dynamic layout is depicted in Figure 3.15. It shows how the word "*sky*" can be typed on such keyboard. Note that the layout is rearranged in each step.

3.3.1 Static distribution

Static distributions are not as cognitively demanding as dynamic distributions as the users may memorize the character distribution relatively easily. Some of the methods can be even eyes-free for an expert user, depending on the interaction modality used.

A static distribution is designed prior to using the text entry method usually by analyzing static frequencies of single characters or *n-grams* (i.e. n successive letters). The distribution can be either a result of designer's creativity or computed by an algorithm. Finding the distribution is an NP-complete optimization problem [91] and thus either a search heuristic or search space constraints needs to be used. A common search heuristic is a genetic algorithm used for layouts of ambiguous keyboards (see Section 3.4.3), e.g. [65, 47, 81]. Yin and Su [222] used a particle swarm optimization algorithm for the same purpose. Search space constraints were used in ambiguous keyboard design [47, 104] by forcing the layout to follow the alphabetical order.

3.3.2 Dynamic distributions

In dynamic distributions, the sequence of user inputs to enter a character is changed gradually according to the actual probability in current context [16]. The distribution is usually computed from a letter-level language model (see Section 3.4.2 for more details). This approach minimizes the number of user inputs needed to enter a character at the expense of higher cognitive demands on the user.

Dynamic distributions are used mostly in scanning systems by rearranging letters. This

can be done either in the whole layout (e.g. [92, 209, 66]) or only locally. The local approach tries to retain letters close to their original positions by rearranging only parts of the layout similarly to aforementioned LetterWise [105] method (e.g. [126, 127, 131]). Another approach is to retain letter position static while making the scanning sequence dynamic (see Chapter 7). Dynamic layouts may offer not only single letters, but also several highly probable continuations (see Chapter 9).

Text entry methods often combine dynamic and static aspects in order to lower the cognitive demands. An example of such combination is Dasher [211]. Although the letters are always displayed alphabetically in this method, their size changes dramatically according to their contextual probability. Another dynamic keyboard GazeTalk [76] shows dynamically six most probable letters. When a desired letter is not shown, a static full keyboard is available. SpreadKey [123] is a virtual QWERTY where low-probable characters close to currently typed characters are replaced by high probable characters. pEYEwrite [201] is a method based on three hierarchical pie menus. While the first two menus are static, the last menu dynamically provides five most probable next letters.

3.4 Use of Language Models

Many different language models are used in wide range of scientific disciplines. Thus, no exact definition of a language model exist. Generally, we may say that language model is a mean for describing languages in a structured and consistent way. In this section, the use of language models in text entry methods is described. We identified two main techniques, which are inevitably dependent on a language model: prediction (see Section 3.4.2) and ambiguous keyboards (see Section 3.4.3). They are not mutually exclusive and can be combined in a single text entry method.

Almost every text entry method inherently uses a language model—from simple static probabilities of characters to more sophisticated language models. While number of language models exists, three essential approaches can be found in the literature on text input [16, 45]: *syntactic, semantic and statistical*.

Syntactic and semantic approaches store rules either in probability tables or as a grammar. The difference between these two approaches lies in categorization of words (syntactic or semantic categorization). The semantic approach was employed, for example, in the work by Demasco and McCoy [28].

The *statistical* approach stores frequencies of *n-grams*. An n-gram is a sequence of n items. The item can be either a letter or a word. Based on the item type we can then distinguish between letter-level n-grams [211] and word-level n-grams (e.g., Google n-gram corpus [48]). N-grams with length equal to one, two, and three are called unigrams, bigrams, and trigrams respectively. The *order of the model* further refers to the longest n-gram contained in the language model. The probability of the next items are extracted from the model based on already written $n - 1$ items. These written items (letters or words) are

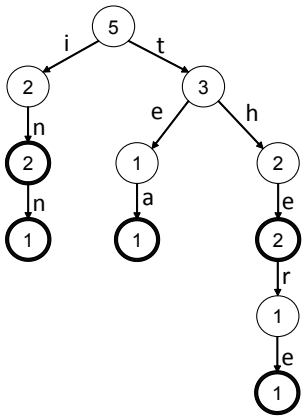


Figure 3.16: An example of a letter-level language model in form of the prefix tree (often referred to as *Trie*). The model contains five words: "*tea*", "*in*", "*the*", "*inn*", and "*there*". The numbers in the nodes express frequency of letters.

usually referred to as context. Even though the statistical approach is the simplest one [45], it prevailed in text entry methods published in the past decade, while the other two approaches diminished.

3.4.1 Statistical Language Models

Text entry methods usually use two types of statistical language models: letter-level and word-level. Obviously, the type of the language model depends on the type of n-grams stored (letters or words).

Letter-level language models are usually stored as a prefix tree, often called *Trie* (from re**trie**val). This data structure enables fast basic operations for n-grams: indexing, adding, and deleting. An example of a Trie is shown in Figure 3.16. The 1^{st}-order model stores only unigrams. It can be used for statically distributed layouts (see Section 3.3.1) as only static probability of letters is stored. In order to employ dynamic layouts and prediction, a higher-order model has to be involved. Several orders of the models are reported in the literature, for example, 2^{nd}-order [201], 3^{rd}-order [17], 4^{th}-order [105, 126], 6^{th}-order [159, 158], or 8^{th}-order [142]. With increasing order of the model, the performance of the prediction improves. However, the performance does not improve significantly after the 6th-order model (see Section 9.5).

An interesting system based on letter-level statistical approach is described by Shieber and Baker [171]. In this system, words are completed from abbreviations—the user omits all vowels and consecutive duplicate consonants while typing. This approach saves about 30% of keystrokes.

The basic model described above works well for previously seen n-grams stored in the model. However, a text entry method often has to predict probabilities of all characters in

given context including n-grams that have not yet been stored in the model. This is called zero-frequency problem [215] and can be solved using one of zero-frequency estimation predictor from adaptive data compression techniques. For example, a predictive text entry method Dasher [211] uses prediction by partial match algorithm [192] to deal with the zero-frequency problem.

Word-level language models are usually stored in tables [45]. The order of the model is rather limited because of high storage complexity. For example, a 3-gram dataset from Google n-gram corpus [48] contain about 23×10^9 entries. Even though the n-grams can be limited to only the most probable, text entry methods usually use only unigrams or bigrams.

3.4.2 Prediction

A text entry method can be accelerated by prediction, when a list of completions is updated with each entered character. The aim of a predictive system is to reduce number of keystrokes per character by offering shortcuts to the most probable characters or words. The ranks of the offered entries are determined in the language model.

Prediction has been heavily used in the text input for motor-impaired people. One of the first prediction system, used in the Reactive Keyboard [26], predicted possible words according to the 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 example is VITPI system [15] which offered unambiguous parts of words found in a dictionary. In Dasher [211], letter-level prediction was used to alter size of virtual keys. Combination of letter-level and word-level language model was used in GazeTalk [76], which predicted six most probable characters and six words according to the current context. Prediction of longer chunks of text is studied in this thesis in Chapter 9.

3.4.3 Ambiguous Keyboards

The basic idea of ambiguous keyboards is to divide the alphabet into several groups of letters and then to assign each group to one key. The ambiguity lies in the fact that a group of letters is assigned to one key only. In order to select the desired letter from the group, a disambiguation process has to be executed. Two disambiguation processes are commonly reported in the literature: letter-level and word-level disambiguation.

Letter-level disambiguation

The letter-level disambiguation is used in the popular mobile text entry method—Multi-Tap and its variations [147, 105]. In this method, the user first selects the desired key and then disambiguates the character by multiple keystrokes. The character is then entered after a timeout is reached or another key is pressed. From the character selection point of view, the technique corresponds to two successive selections: a direct selection followed by step scanning (see Section 3.2.2).

In text input for motor-impaired people, the two selection in the disambiguation technique can be replaced to lower the number of required input signals or to better fit to the interaction modality used. For example, in work by Mirro-Borras et al. [127, 126, 128, 129, 130], the selection process is done as follows: first, automatic selection is employed to select the key and then inverse scanning is used while keeping a switch activated. Note, that after replacing the selection process by two successive steps of automatic scanning, the letter-level disambiguation turns into row-column scanning (see Section 3.2.2 for more details). The letter-level language model is used in their work to shuffle characters on individual keys according to probability, similarly to the LetterWise method [105].

Word-level disambiguation

The word-level disambiguation was first used in the commercial T9 system by Tegic Communications [52]. This method was widely used in mobile phones with 12-button keypad. Instead of entering a particular letter, the user selects corresponding key which contains the letter. After entering a sequence of keys, the list of most probable words is shown. The list is a result of a disambiguation process which selects possible words from a word-level language model. The model is often referred to as a dictionary. This is the strength as well as weakness of the method. Thanks to the dictionary disambiguation process, the method is very efficient in terms of keystrokes per character (KSPC). According to MacKenzie [99], the theoretical KSPC value is very close to 1. This value, however, does not include non-dictionary words. When the user wants to enter a new word which does not exist in the dictionary, another method has to be used, which significantly slows the typing and increases the KSPC rate. The efficiency of the method is heavily dependent on quality and completeness of the dictionary [53].

Many ambiguous keyboards were designed for physically impaired people. Kushler [88] 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 et al. [191] published a similar system, in which only four physical keys were used. Besides disambiguation, the text entry method was capable of predicting words. Harbusch and Kühn [65] presented a similar method, in which the whole alphabet was assigned to only three keys and one key was used for executing a special command in a menu.

Scanning ambiguous keyboard [104] is an ambiguous keyboard with word-level disambiguation, in which the direct selection of a key is replaced by automatic scanning. Because of its efficiency with minimum input signals, scanning ambiguous keyboard became quite popular among text entry methods for motor-impaired people. For example, Kühn and Garbe [86] used a four-key scanning ambiguous keyboard. Harbusch and Kühn [64] showed that the scanning ambiguous keyboard outperforms other scanning text entry methods. Belatar and Poirier [12] used three keys and developed a virtual mobile keyboard. In Qanti [34] three keys are mapped to the alphabet.

3.5 Interaction Modalities

Number of different input modalities can be used to help motor-impaired people interact with a computer. This section summarizes the most prevalent modalities for the text input. Some text entry methods are designed generally to a certain extent—they can use multiple modalities which share a certain feature. An example is the MDITIM method [74] designed for any device which can yield four direction signals (e.g., arrows, joystick, game controller, trackball). On the other hand, some text entry methods are tailored to a single modality only (e.g. [210, 132]).

People with residual dexterity in hands can still use a PC keyboard. However, they usually have problems with chording input (pressing two or more keys simultaneously). This problem is solved by accessibility option in operating systems (e.g., Sticky keys in Windows). Several special devices have been developed to decrease motoric demands on the user, for example, Maltron ergonomic keyboards[1]. Chubon and Hester [24] presented an optimized PC keyboard for typing with one finger only by rearranging letters. Felzer et al. [35] described a keypad with 4×5 layout which used an on-screen keyboard to enable typing.

Scanning is operated by a switch as already mentioned in Section 3.2.2. Number of different switches exist, such as mechanical switches (or buttons), pneumatic switches controlled by breath, bite switches etc. An exhaustive list of switch types is given in work by Ntoa et al. [140]. Some non-standard modalities have been used as switch, for example, eye blinks [8, 166], eye movements [115], intentional muscle contractions [34], or non-verbal vocal input (see Chapter 7).

Game controllers were found to be ergonomic not only for gamers, but also for people with motor impairments. For example, MDITIM method [74] can be controlled with the joystick. Dual joysticks, which are present on some game controllers, can be used to improve text entry rate [214, 84]. Joystick text entry with prediction for motor-impaired people was studied in work by Song [177]. Felzer and Rinderknecht [40] used an 8-way direction pad on a game controller to enter a text. Another use of a game controller is reported in H4-Writer [107] and in Continuous EdgeWrite [120].

For motor-impaired people, the mouse is often replaced by another pointing device like

[1] `http://www.maltron.com/`

trackballs, head tracking, or gaze interaction. Trackballs were used in several text entry methods for a gestural input [74, 217] or on-screen keyboards (e.g., SpreadKey [123]). Head tracking is a popular mouse substitute for quadriplegic people and many commercial products already exist (e.g., SmartNav by NaturalPoint[2]).

3.5.1 Gaze interaction

Gaze interaction is based on an eye tracker device which computes the focal point by measuring eye positions. This interaction has been used in many virtual keyboards. The absence of an explicit selection technique in the gaze interaction has been mostly solved by the dwell-time approach or multimodal interaction (see Section 3.2.3 for further detail).

Using dwell time, typing was found slower than mouse but comparable to head tracking [58]. The dwell time can be adjusted to gain maximum speed [113] and the feedback can be improved by visual or audio cues [114]. The accuracy of typing can be improved by prediction [111]. The dwell time, entry rate, error rate, and workload was analyzed in an extensive study by Räihä and Ovaska [156]. Špakov and Majaranta [180] presented a scrollable keyboard which occluded minimal screen space. Panwar et al. [145] described an optimized design based on Fitts' law [42, 101].

Midas touch problem was solved by Isokoski [73] by using off-screen targets adjacent to the screen. The dwell time can be avoided either by context switching between two on-screen keyboards [132] or by using prediction [85].

The selection by other modality includes selection by smiling [200], tooth clicking [225], or speech recognition [11], in which users pronounce a letter and look at it at the same time. The eye trackers are usually very expensive, however, several low-cost devices exist which can be used for text input [163, 10]. Gaze gestures were used, for example, in EdgeWrite [220] or Dasher [199]. Urbina and Huckauf [201] presented selection by gaze gesture in hierarchical pie menus.

3.5.2 Acoustic modality

Acoustic input modality can be also used for entering text. Many automatic speech recognition (ASR) systems for motor-impaired people support entering text by dictation in word-by-word manner [139] or letter-by-letter manner (e.g., [117, 118, 138]). An interesting combination of ASR and continuous gesturing was presented in Speech Dasher [205]. Motor-impaired people reach similar performance to able-bodied people with an ASR system [167]. However, users with speech impairments (e.g., dysarthria) report poor accuracy of ASR systems. Sometimes, it can be partially solved by tedious training of an ASR system or by customizing keywords [56]. The accuracy still remains lower when compared to people with clear speech.

[2] `http://www.naturalpoint.com/smartnav/products/4-at/`

For people with combined motor and speech impairment, the solution is the non-verbal vocal input (NVVI). In this interaction modality, other sounds than speech are used like whistling, humming, hissing, etc. NVVI has been already used for text input in work by Sporka et al. [181]. The method is described in detail in Section 2.3.1. Other text entry methods operated by humming and hissing are described in this thesis in Chapters 7–9.

3.5.3 Bio signals

Another technique, which can be used in text input, is measuring bio signals. Two main non-invasive modalities belong to this technique: *intentional muscle contractions* and *brain-computer interaction*. Intentional muscle contractions are measured as peaks in EMG (electromyography) signal which is produced by skeletal muscles. The electrodes are attached onto skin of the user. For example, Felzer et al. [33, 34] attached the electrode to forehead and the switch was triggered by raising the eyebrow.

The brain-computer interaction usually use EEG (electroencephalography) signals recorded along the scalp. The process of decision making in the human brain yields a P300 wave. When the P300 wave is detected, a switch is triggered. EEG uses more expensive hardware than EMG and the signal processing is more complex but it can help paralyzed users with a locked-in syndrome.

Text input based on EEG was used in P300 speller [210]. 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. In RSVP [66, 142] the letters are presented one after another sorted according to the probability in a letter-level language model. The speed of these keyboards is still quite slow, 1.5 WPM for P300 speller and 2 WPM for the RSVP were reported.

3.6 Experimental Setup of Text Entry Methods Evaluations

Every new text entry method has to be thoroughly evaluated to assess its possibilities and limitations. The experiments described in literature often vary greatly in methodology, design, and evaluation. No standardized experimental setup exists but the one described in ISO 9241-4. It is unfortunately designed especially for evaluation of a standard computer keyboard, not universally for a common text entry method. Therefore, almost no paper follows this experiment. Number of best practices for an evaluation of a text entry method is described in the work by MacKenzie [102].

A common feature of text entry methods is relatively slow learning by the users. For example, learning a new PC keyboard layout takes more than 100 hours [174]. Thus, it is quite common that the evaluation is spread into several sessions to grasp at least the beginning of the learning process. The speed of expert and novice users usually varies significantly.

3.6.1 Basic Experimental Setups

Based on a review of literature related to text entry methods for motor-impaired people, we found that five basic experimental setups are mostly reported: simulation, evaluation with able-bodied people, evaluation with the target group, longitudinal evaluation, and expert evaluation.

Simulation is referred to as a process in which user input is replaced by an algorithm (e.g. [100]). The algorithm simulates user interaction with the examined method by computing a theoretical performance value, such as number of keystrokes required to enter a portion of text. Simulation may be used for theoretical comparison among several layouts in terms of keystroke saving rate (KSR, e.g. [17]). From these values, type rate can be estimated (e.g., [127, A8]). Simulation is essential for computing an optimal letter layout (see Section 3.3.1). Evaluation by simulation is relatively fast and it is convenient for initial design comparisons and assessments. However, it does not take into account human factors such as immediate usability, errors produced, number of foci of attentions, visual search demands, and interaction demands. The simulation is error-free and thus it models performance of an expert rather than a novice user.

Evaluation with able-bodied people is more accurate than simulation as it takes into account some human factors. This experimental setup is usually used for comparison among several designs. This experimental setup often yields quantitative and qualitative results. However, neither qualitative nor quantitative results can be generalized for motor-impaired people. The qualitative results lack the necessary insight and it is not guaranteed that the examined method would pose an asset for the target group. The quantitative results are often significantly different in terms of type rate. For example, Vigouroux et al. [206] found that participants with spinal muscular atrophy are approximately 40-50% slower than able-bodied when typing on an on-screen keyboard. On the other hand, evaluation with able-bodied people can still be used as a proof of concept [104].

Evaluation with the target group is used as a validation of the studied text entry method for the target group. It is mostly held in form of case studies. The results are mostly qualitative providing a deep insight. The reported case studies include only several participants as the study organization with the target group is usually too expensive due to challenging recruitment and transportation difficulties. This is one of the reasons why this kind of evaluation is missing in many scientific papers.

Longitudinal evaluation refers to an evaluation when performance is measured in number of sessions (usually 10 and more). This type of evaluation gives the opportunity to model the performance according to power law of practice [134] and predicts the performance of an experienced user.

Expert evaluation is a short report of peak performance of evaluated text entry method. The expert is an experienced user of the method but in some studies the designation "expert" is attributed to a person with few hours of experience with the method. The performance of an expert user is usually reported in order to express an approximation of

upper limit of the method's type rate.

Experimental setups listed above usually provide different results. For example, type rate of an expert user is much faster than type rate obtained in a case study with a disabled participant. However, the reported values are affected by other variables such as instructions given to participants, length and number of session, choice of phrases to copy, possibility of error correction, etc.

3.6.2 Measures

While type rate is considered as the most important measure, a number of measures expressing other properties of the examined method are reported in the literature. Most of them are described in the work by Wobbrock [216]. Measures can be classified from several points of view:

- *Aggregate / character-level.* Aggregate measures summarize the performance of a method as a whole. The most common are type rates, error rates, and efficiency measures. Character-level measures express performance for each character (e.g., letter confusion matrix).
- *Method-agnostic / method-specific.* According to [216], method-agnostic measures can be used for variety of methods, while method-specific measures for one class of text entry methods only.
- *Absolute / relative.* Relative measures express a relation between two methods, for example, keystroke saving rate (KSR) [17] or selection savings [28]. Relative measures are used for comparisons. Absolute measures can be used to express performance of one method independently.
- *Empirical / theoretical.* Empirical values are measured in an experiment with people, while theoretical values are product of simulation, analysis, or estimation (e.g., KSPC [99]). Theoretical values usually do not take into account errors produced by the user. Theoretical and empirical values differs mostly for novice users. For example, KSPC for the MultiTap method [147] is 2.0 theoretical and 2.2 empirical.
- *Subjective / objective.* Objective data measures performance of the text entry method, while subjective expresses opinion of the users. Subjective data can be further divided to qualitative and quantitative. Qualitative regards to immediate usability [109], participant comments, and researcher observations during the evaluation. Subjective quantitative data are usually obtained from a Likert scale involving several Likert items [93].

Type rate is mostly reported in terms of word per minute (WPM) where a "standard word" is defined as a string of five characters including space. Other similar metrics are characters

per minute (CPM), characters per second (CPS), or seconds per character (SPC). These metrics are easily convertible as shown in the following equations:

$$WPM = \frac{CPM}{5} \tag{3.1}$$

$$CPM = CPS \times 60 \tag{3.2}$$

$$SPC = \frac{1}{CPS} \tag{3.3}$$

Reporting *error rate* is much less standardized. Three favorite error rates exist: keystrokes per character (KSPC) [99], MSD error rate [178], and a unified error metric [179]. KSPC expresses errors that were corrected during the experiment, while MSD error rate corresponds to errors left in the transcribed text. The unified error metric provides similar measures: uncorrected and corrected error rate. The advantage of the unified error metric is the capability to report total error rate as the sum of uncorrected and corrected rates. KSPC and MSD error rates cannot be combined this way [179].

Several other error rate metrics are reported in literature as well: wrong characters rate (e.g. [121], see Equation 3.4), overproduction rate (e.g. [57], see Equation 3.5), or word error rate [205]. Character-level errors are usually presented by a confusion matrix (e.g. [30]).

$$\textit{wrong character rate} = \frac{\textit{\# of wrong characters}}{\textit{\# of total characters}} \tag{3.4}$$

$$\textit{overproduction rate} = \frac{\textit{\# of total keystrokes}}{\textit{\# of optimal keystrokes}} \tag{3.5}$$

3.7 Overview and Evaluation of Text Entry Methods

The examined text entry methods for motor-impaired people are summarized in Table 3.1. Each row corresponds to one scientific publication from year 1988 to year 2013. Except year of the publication, name of the method (empty if not found), and reference, the columns contain following information:

- *Selection type* (see Section 3.2): D—direct selection, S—scanning, P—pointing
- *Character layout* (see Section 3.3): S—static layout, D—dynamic layout
- *Use of language model* (see Section 3.4): Pl—letter-level prediction, Pw—word-level prediction, Al—ambiguous keyboard with letter-level disambiguation, Aw—

ambiguous keyboard with word-level disambiguation

- *Interaction modality* (see Section 3.5): ASR—automatic speech recognition, EEG—electroencephalography, EMG—electromyography, G—gaze interaction, GC—game controller, HT—head tracking, J—joystick, M—mouse, K(n)—n keys, NVVI—non-verbal vocal input, P—a pointing device, PB—push button, S—a switch, TB—trackball, TP—touchpad
- *Evaluation type* (see Section 3.6): A—able-bodied participants, D—disabled participants (target group), L—longitudinal, S—simulation

3.7.1 Overview of Text Entry Methods

In order to select the methods presented in the Table 3.1, we conducted a systematic literature review. In the first step, the publications were collected by keyword search in the main databases like ACM digital library[3], Springer link[4], IEEE Xplore[5], etc. Then, we examined their references and citations and added publications focusing on text input for motor-impaired people. This process was done recursively, until no new publication was added. From the obtained pool of publication we selected 61 which reported on an original text entry method. The rest of publications were mostly studies of already existing methods.

Even though the type rate in WPM can be found or computed in most evaluations (49 out of 61 publications), we cannot overestimate this number. Not only because of influences in the experimental setup (kind of participants, number and length of sessions, instructions, phrases to copy, ...) but also because of the way researchers report the type rate. Publications often state three different values. The most common one is the arithmetic mean measured in the last session. Another one is the grand mean which equals to arithmetic mean measured across all session. Some publications also report the peak type rate achieved. Although all these values are used to express the type rate of the method, they are obviously quite different. Therefore, the Table 3.1 does not show only one WPM value. It rather gives a range of WPM values expressing lowest and highest WPM values as reported in publications.

In the summarized text entry methods for motor impaired people, selection by scanning is used in 27 of them, pointing in 24, and direct selection in 16. Note that the total number does not equal to 61 as several papers describe more than one method design and those designs can have different properties. Unsurprisingly, scanning is slowest input technique as only one switch is used (WPM mean = 3.75±2.5, median = 3.5) when compared to average method using direct input (WPM mean = 10.1±9.5, median = 6) or pointing (WPM mean = 12.0±10.5, median = 10).

[3] `http://dl.acm.org/`
[4] `http://link.springer.com/`
[5] `http://ieeexplore.ieee.org/`

Year	Name & Ref.	Selection	Layout	Use of LM	Modality	Evaluation	WPM
1988	Chubon [24]	D	S		K(26)	A	
1988	*P300 speller* [31]	S	S		EEG	A	0.5
1992	*Reactive Kbd.* [26]	D	S	Pw	K(26)	A	
1993	*Half-QWERTY* [121]	D	S		K(13+1)	A L	23 - 42
1998	Jones [77]	S	S	Pl Pw	S	A	
1998	Kushler [88]	D	S	Aw	K(8)		
2000	*GRAFIS* [6]	S P	S	Pw	S/M		
2000	*MDITIM* [74]	D	S			A	3 - 10
2000	*Dasher* [211]	P	D	Pl	P/M/HT/G	A	18 - 34
2001	*UKO* [86]	D	S	Aw	K(4)	D	6
2001	*SUITEKeys* [118]	D	S	Pw	ASR		16
2002	*TouchMeKey4* [191]	D	S	Pl Pw	K(4)	S A	14 - 23
2003	*GazeTalk* [57]	P	D	Pl Pw	P/M/G	A	4 - 6
2003	*UKO-II* [65]	D S	S	Aw	K(?)		
2003	*EdgeWrite* [219]	P	S		TP	A D	6 - 7
2004	Evreinova [30]	D	S		K(4)	A	15
2006	Baljko [9]	S	S		K(1+1)	S A	1.8
2006	*LURD-Writer* [36]	P	S	Pw	EMG	D	1 - 2
2006	Sporka [181]	D	S		NVVI	A	2 - 3
2006	*EdgeWrite* [217]	P	S		TB	A D	7 - 12
2007	*SEK* [82]	P	S		HT/M	A D	2 - 3
2007	Lin [94]	S P	S			D	
2007	*AUK* [133]	D S	S		K(1-10)		
2007	Norte [137]	S P	S		M/K(1)	D	
2008	*HandiGlyph* [12]	S	S	Aw	PB	D L	2 - 3
2008	Lin [95]	S	S			A	1 - 2
2008	MacKenzie [111]	P	S	Pl Pw	G	A	10 - 13
2008	Miro-Borras [126, 127]	S	D	Al	PB	S	
2008	Spakov [180]	P	S		G	A	7 - 16
2008	*Dasher* [199]	P	D	Pl	G	A L	15 - 20
2008	Sibylle [209]	S	S	Pl Pw	PB		
2008	*EyeWrite* [220]	P	S		G	A L	2 - 8
2009	*3dScan* [39]	S	S		EMG	D	1 - 2
2009	MacKenzie [103]	S	S	Aw	K(1)	A	4 - 6
2009	Majaranta [113]	P	S		G	A L	16 - 22
2010	*BlinkWrite2* [8]	S	S	Aw	G	A	4 - 6
2010	*Qanti* [34]	S	S	Aw	EMG	A D	2 - 7
2010	*SAK* [104]	S	S	Aw	K(1)/EMG	S A D	2 - 8
2010	*SpreadKey* [123]	P	D	Pl	TB	S D L	13
2010	Miro-Borras [129]	S	S D	Al Aw	K(1)	S A	6 - 16
2010	*KKBoard* [132]	P	S		G	A	10 - 20
2010	Roark [159]	S	D	Pl	PB	A	2 - 5
2010	Song [177]	S	S D	Pl Pw	J	A	5 - 8
2010	*pEYEwrite* [201]	P	S D	Pl	G	A L	10 - 17
2010	*Speech Dasher* [205]	P	D	Pl	ASR/G	A	40 - 54
2011	*CHANTI* [A6]	D	S	Aw	NVVI	D	2 - 5
2011	*Humsher* [A4]	D S	D	Pl	NVVI	A D	3 - 6
2011	*RSVP* [66]	S	D	Pl	EEG	D	
2011	*BlinkWrite* [166]	S	S	Aw	G	A	4 - 5
2011	Prabhu [152]	S	S		K(1)	S A	1.3
2012	Beelders [11]	P	S		ASR/G	A	2 - 4
2012	*DualScribe* [35]	D	S	Aw	K(18)	S A D	3 - 5
2012	Lafi [89]	S	S		K(1)		
2012	*Cont. EdgeWrite* [120]	P	S	Pw	GC	A	5
2012	*RSVP* [142]	S	D	Pl	EEG	A D	1.8
2012	*EyeBoard* [145]	P	S		G	A	4 - 5
2012	Raiha [156]	P	S		G	A	20 - 24
2012	Zhao [225]	P	S		G	A	8 - 12
2012	Roark [158]	S	D	Pl	K(1)	A	2 - 5
2013	Tuisku [200]	P	S		G	A	3 - 4

Table 3.1: Summary of methods. Please see Section 3.7 for the explanation of abbreviations.

Scanning ambiguous keyboards appear to be fastest among scanning methods. The aggregated WPM values for scanning ambiguous keyboards are WPM mean = 5.3±2.6 and median = 5, while the rest of simple scanning methods yield WPM mean = 2.6±1.7 and median = 1.8.

Static layouts are highly preferred (50 publications) to the dynamic layouts (14). However, we did not find any trend in terms of type rate when comparing dynamic and static layouts.

Regarding the use of language models, prediction was used in 16 methods on the letter level and 12 methods on the word level. The disambiguation is popular on word level (11 publications), while letter-level disambiguation is reported only in two methods.

Modalities used in the methods are summarized in Figure 3.17. Note, that most methods use some kind of keyboard or keypad, usually restricting to only several buttons or keys. A lot of papers do not describe the actual interaction modality and specify only number of inputs.

3.7.2 Evaluation of Text Entry Methods

Evaluation by simulation have been done in 9 publications. An evaluation with able-bodied or disabled users is present in 53 out of 61 publications. The average numbers of participant count, session count, and session length are summarized in Table 3.2 separately for able-bodied and disabled participants. These number are shown together with the number of publications that contained the required information.

Note, that evaluations with disabled participants were done in much less cases (40 vs. 18) and with less number of participants. Average number of sessions is similar in both cases, but session length is somewhat longer in case of disabled participants. Those numbers correspond to the premise that evaluation is usually quantitative with able-bodied participants and quantitative with disabled participants as described in Section 3.6.1. Reported type rates of disabled participants (WPM mean = 4.5±3.2, median = 4) are slower than those of able-bodied (WPM mean = 9.3±9.6, median = 5). This is partly because publications with high-speed methods often lack evaluation with disabled participants (see Table 3.1), and partly because they usually achieve slower type rates than able-bodied participants [206].

As shown in Figure 3.18a, the type rate is mostly reported in the word per minute (WPM) rate (32 evaluations) and characters per minute (CPM) rate (10). Other metrics of type rate are used only marginally. Note that slower methods are more likely to be reported in CPM. For CPM reports the mean type rate is 3.5±2.6 (median = 3.5), and for WPM the mean is 10.9±10.0 (median = 6.5).

Error rate metrics are much more diverse as already mentioned in Section 3.6.1. Their use depends on the experiment setup. For example, MSD error rate (i.e., errors left in the transcribed text) is useless when error corrections are not allowed within the text-copy task. Figure 3.18b shows number of error rate metrics reported in the literature. The

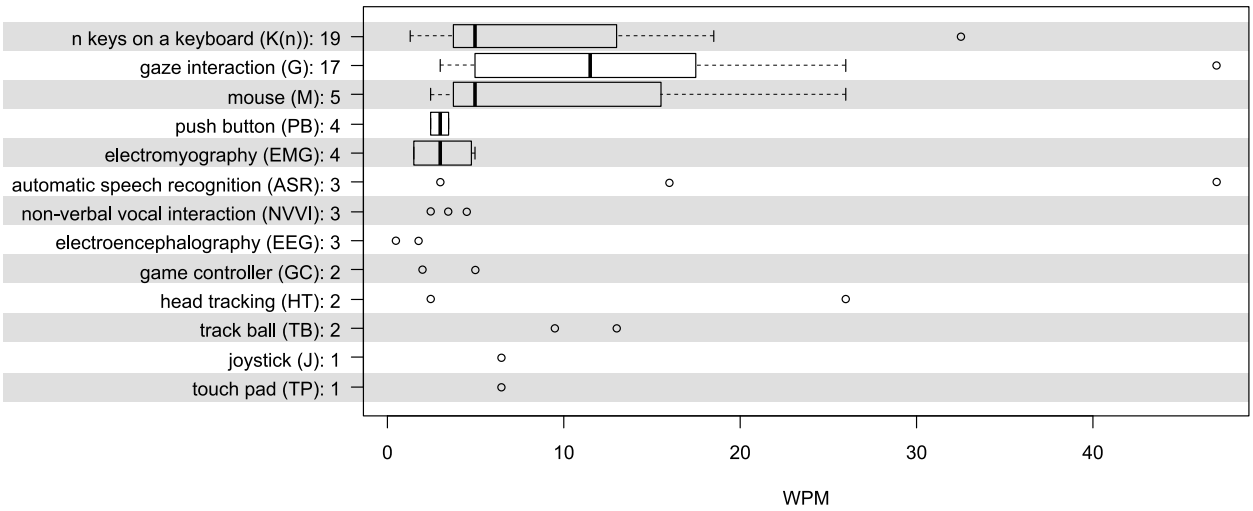


Figure 3.17: Boxplot of WPM of the methods aggregated by modalities. Labels contain name of the modality, abbreviation, and the number of corresponding publications.

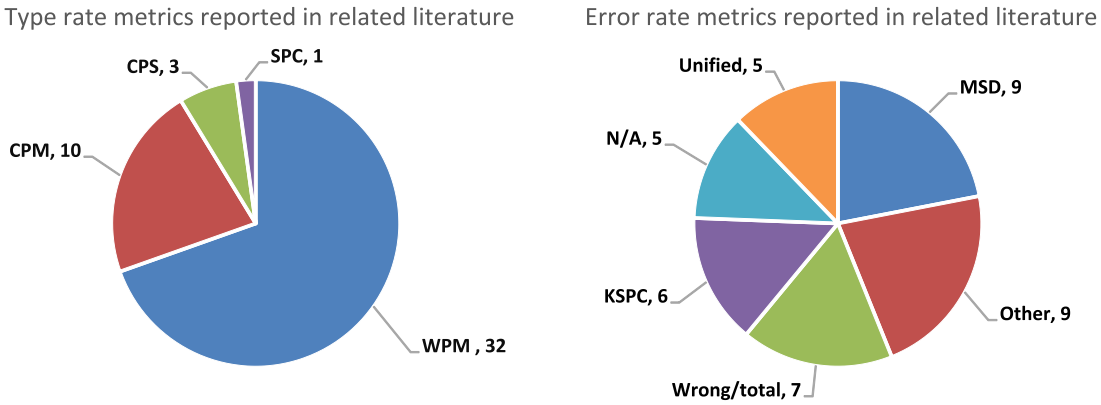
	Able-bodied participants		
	Mean (SD)	Median	# of pub.
participant count	8.8 (±4.0)	8.5	40
session count	5.3 (±4.7)	4	39
session length	43 (±24.5) minutes	50 minutes	17

	Disabled participants		
	Mean (SD)	Median	# of pub.
participant count	2.5 (±2.7)	1	18
session count	5.4 (±4.6)	5	11
session length	53 (±15) minutes	60 minutes	4

Table 3.2: Summary of published evaluations of the methods in terms of participant count, session count, and a typical session length. *# of pub.* is the number of publications containing the required information.

# of pub.	Likert item
3	speed, method fatigue
2	ease of use, modality fatigue, efficiency
1	stress, frustration, comfort, satisfaction, difficulty, accuracy, enjoyableness

Table 3.3: Summary of Likert items reported in the literature.



(a) Summary of type rate metrics.

(b) Summary of error rate metrics.

Figure 3.18: Summary of type rate and error rate metrics as reported in the literature.

most widely used are MSD error rate (reported in 9 publications), wrong characters rate (7), KSPC (6), and unified error rate (5). Nine publications report other metrics, such as overproduction rate, confusion matrix, or word error rate. In other nine publications, error rate is reported but the method of calculation is not stated.

Subjective data obtained by Likert items are reported in nine papers. They focus mostly on perceived speed, method/modality fatigue, ease of use, and efficiency. The full list of Likert items with number of occurrences in publications is shown in Table 3.3.

3.8 Summary

The aim of this chapter is to describe the research related to text input for motor-impaired people. We discuss and give an overview of techniques for letter selection, possibilities of letter distributions, prediction methods, and interaction modalities. We also describe typical experimental setups and metrics which assess performance of a text entry method. The methods found in the related scientific literature are summarized and classified according to the aspects listed above. The approximate type rate of each method is also shown and implications are discussed.

Text entry for motor-impaired people is much slower than typing on a PC keyboard by an able-bodied person. The extreme case is scanning, which is used when only one input is available. Scanning ambiguous keyboards show the best performance among the scanning text input. Although static layouts are highly preferred in the literature, their speed is comparable to dynamic layouts. Some dynamic layouts even outperform the static ones on the expense of escalated cognitive demands. As the text input for motor-impaired people is slow, prediction is popular technique to improve the text entry rate.

Evaluation of text entry methods is still mostly done with able-bodied as a proof of concept.

However, disabled participants usually achieve slower type rates in the experiments. The type rate is mostly reported in words per minute (WPM), but slower methods are more likely to be reported in characters per minute (CPM). No consensus exists on error rate metrics. Subjective data are obtained in qualitative or quantitative manner. Popular are Likert items focusing mostly on perceived speed, method/modality fatigue, ease of use, and efficiency.

Comparing different text entry methods is rather complicated as experimental setups varies in each publication. Although ISO 9241-4 describes a standardized experimental setup, no reported evaluation follows this setup. We believe that it is because the ISO 9241-4 experiment setup is tailored for manufacturers of computer keyboards. Thus, we see a need for a standardized experimental setup which would apply to a general text entry method as a future work. Defining such setup should be a result of a broad academic discussion as we need to find the tradeoff between resources required and the validity of results. In order to improve comparability of the methods, the standardized experimental setup should define namely the following:

- *Procedure* including minimal number of participants, number of sessions, and length of one session or rather length of each experimental condition in one session. These three numbers should be balanced in order to yield a reasonable amount of hours to conduct the experiment.
- *Apparatus* including phrases to copy, error correction capability, and error feedback if corrections are allowed. Participant instructions should be also standardized.
- *Dependent variables* including type rate and error rate reporting. Standardized post-test questionnaire should be also developed.

On the other hand, number of conditions should not be defined as often more methods are needed to be tested. However, general guidelines can be applied such as choice of the correct baseline, keeping the number of conditions low, and ensuring correct counterbalancing.

When looking on the basic experimental setups, we see a potential in standardizing two of them: the evaluation with able-bodied people and the longitudinal evaluation. Simulation is often too much dependent on the entry method to be standardized and expert evaluation is reported only rarely. We also do not see such potential in the standardization of an evaluation with disabled participants. The first reason is that the group of motor-impaired people is too diverse and it is often difficult to find two people with the exactly same conditions. The other reason is that organizing a quantitative experiment with motor-impaired people would be too challenging and expensive. Therefore, qualitative case studies are preferred by the researchers.

Part II

Contribution

4 Overview of contributions

This chapter describes limitations of the state of the art and introduces the main contributions of the thesis.

4.1 Limitations of the State of the Art

The previous two chapters thoroughly describe the state of the art of non-verbal vocal input and text input for motor-impaired people. The NVVI has been studied so far only in one method—direct selection by encoding on a QWERTY keyboard [181] in this context. For motor-impaired people, however, more appropriate text entry techniques exist such as scanning, ambiguous keyboards, or predictive completion.

The continuous input of NVVI has been studied extensively (e.g., [185, 184, 14, 61]), while event-based input was mostly neglected. However, the event-based input fits better for the task of text entry. Moreover, continuous input is more demanding for the vocal folds, as the user might be required to hold the tone for quite a long time.

From the literature survey, it is known that hissing and timbre-based input can be used by motor-impaired people [3, 63, 62]. However, the usability and applicability of the pitch-based input has not been previously evaluated with motor-impaired people and only proof-of-concept studies with able-bodied people has been published. Moreover, different vocal gestures have been used so far, but very little is known about the acceptability of different gestures by different target groups (able-bodied, motor-impaired, speech-impaired people).

4.2 How this Thesis Extends the State of the Art

The thesis contributes to the field of human-computer interaction (HCI). The HCI is a discipline, which studies design, implementation, and evaluation of interactive systems from the perspective of use by people. The HCI research field originated in 1980's and is steadily growing since then.

The primary subject of research in the HCI field is the interaction of *people* with *computers*. Computers are studied within the computer science, which stems from formal technical sciences. People, on the other hand, are mostly studied in human-oriented social sciences. The most important social sciences used in the HCI field are psychology, sociology, communication studies, and cultural studies. Some researchers see the HCI as a "bridge" connecting psychology and informatics [68]. In accordance with this view, the thesis includes contributions of both HCI aspects: technology-oriented and human-oriented.

4.2.1 Technology-Oriented Contributions

The technology-oriented contributions of this thesis are outlined below, namely audio processing for NVVI, novel text entry methods, and models of the methods.

Novel text entry techniques. Several novel text entry methods controlled by NVVI are presented in this thesis. They have two aspects in common. They are all operated by NVVI (humming and hissing) and they all utilize a letter-level prediction. Two methods are described in Chapter 7 and use scanning exclusively: the *N-ary scanning* and *row-column scanning on an array*. Static layout arrangement and prediction are used in both methods, which is achieved by introduction of dynamic scanning.

Other novel text entry method is Humsher (Chapter 9), which includes four different interfaces. These interfaces are capable of predicting chunk of letters and the selection method combines direct selection and scanning. Further, an ambiguous keyboard (CHANTI) with word-level prediction is studied in Chapter 8. Although the ambiguous keyboard is not novel itself, the presented evaluation with disabled people brings novel insights.

Models of selected text entry methods Models of the N-ary scanning method (Chapter 7) and Humsher (Chapter 9) are described. These models are capable of estimating entry rate for an expert user and thus define theoretical upper limit of the method. The models are evaluated with users. A strong correlation between simulation and experiment values indicates the validity of the models. The model for the ambiguous keyboard is not described as it has been already described in previous work [104]. Based on the proposed model of Humsher, we also investigated in an optimal size of language model for letter-level prediction.

Audio signal processing for non-verbal vocal input. In order to build an NVVI application, a proper technical solution is needed for the input sound processing. One of the approaches to improve robustness of the sound processing is the inclusion of a classifier, which selects the segments of non-verbal parts in the input audio signal. The segmentation method is introduced in Chapter 5 and is based on computing mel-frequency cepstral coefficients processed in a neural network classifier. Further comparison with the already existing segmentation method results in significantly improved accuracy.

4.2.2 Human-Oriented Contributions

Human-oriented contributions of this thesis include a study on vocal gesture design and investigation in applicability of vocal gestures for disabled people.

Vocal gesture comparison. In order to choose a proper set of vocal gestures for future NVVI applications, a controlled experiment was conducted exploring fatigue, satisfaction, and efficiency of different vocal gestures. The experiment (Chapter 6) was quantitative and was done with able-bodied participants. A set of design guidelines for a NVVI application was derived from the experiment.

Applicability of NVVI for disabled people. Two longitudinal studies were conducted to find the applicability for motor-impaired people. Seven participants with different levels of motor and speech impairments were included in these studies and the interaction by humming with a predictive keyboard (Chapter 9) and an ambiguous keyboard (Chapter 8) were studied. We found that the applicability varies according to the level of impairment and the applicability of NVVI is higher than the applicability of speech recognition. In other words, people who are not able to interact with computer by speech, are still able to use the NVVI. Moreover, we found that the target group of NVVI users can be broaden by careful design of the vocal gestures.

4.3 Summary

The limitations of the state of the art have been described in this chapter together with the main contributions of this thesis. The contributions are further discussed and described in detail in the following five chapters.

5 Segmentation of Speech and Humming

The non-verbal vocal input complements traditional speech recognition systems with continuous control. In order to combine the two approaches (e.g. "volume up, *mmm*"), it is necessary to perform a speech/NVVI segmentation of the input sound signal. This chapter presents a real-time method for segmentation of silence, speech, and humming in an input audio signal. The method is based on classification of MFCC and RMS parameters using a neural network (*MFCC method*). The method is compared to an existing method based on computation of volume changes in the signal (*IAC method*) as presented by Sporka [182]. The results indicate that the MFCC method outperforms IAC in terms of accuracy, precision, and recall. The research described in this chapter has already been published in [A2].

5.1 Motivation

Speech control is not suitable for inputting continuous and real-time data. For example, specifying a position of mouse cursor by speech is rather awkward. Another example, where the speech input fails, is playing computer games that require real-time control [184]. An alternative can be provided by the non-verbal vocal input (NVVI) as described in detail in Chapter 2.

Non-verbal vocal input cannot be considered a replacement for speech interaction, as the expressive capabilities of NVVI are rather limited. However, NVVI complements traditional speech recognition systems by continuous control. Igarashi and Hughes [72] suggested using the length of a tone produced after an utterance to emulate a joystick with immediate feedback. This would be very useful for example for moving a mouse. The user can say "move up, *mmm*" and the cursor moves up while "*mmm*" continues.

Currently, NVVI and speech recognition systems exist separately. In order to combine these two approaches so that the scenarios described above can be implemented, we need a method that analyzes the input audio signal and determines the segments containing verbal utterances and non-verbal vocal gestures. These segments will be further processed by existing speech and NVVI recognizers.

Processing of an audio signal is an integral part of each application, in which NVVI is used. The processing is done in three steps similarly to automatic speech recognition. First, the signal is sampled in a sound card, then features (e.g., pitch) are extracted from the signal and then the vocal gestures are classified and the application provides a corresponding feedback. These phrases are shown in Figure 5.1. However, when combining NNVI and speech for the interaction, the signal has to be segmented in order to process parts of the signal appropriately. This situation is shown in Figure 5.2.

This chapter describes a method of NVVI sounds (humming) and speech segmentation. The method can be used not only for combining NVVI and speech commands, but also for

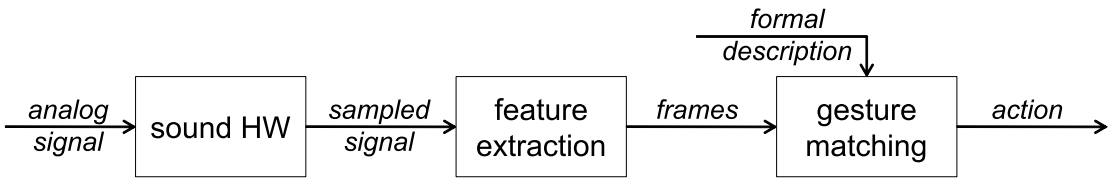


Figure 5.1: Detailed phrases of NVVI signal processing.

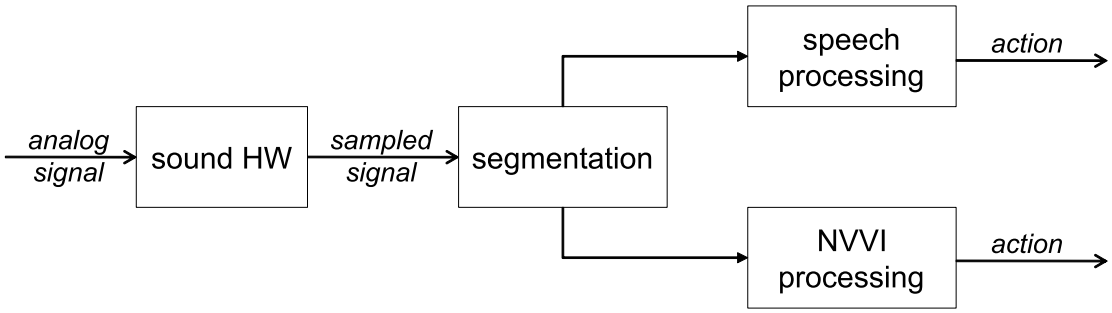


Figure 5.2: Overview of phrases when segmentation is incorporated.

filtering off spontaneous utterances when controlling an NVVI application.

5.2 Related work

Processing vocal input is a traditional area in the field of signal processing. Numerous works have been published in recent years, but most of them concern speech processing, and only a small proportion deal with processing non-verbal sounds. The vital part of each speech recognizer is a speech/non-speech detection that selects parts of an input audio signal to be processed by the recognizer. A considerable amount of work exists on the speech detection. For example, Martin et al. [119] used linear discriminant analysis applied to mel-frequency cepstral coefficients, while Shafran and Rose [169] used non-parametric estimation of the background noise spectrum using minimum statistics of the smoothed short-time Fourier transform. Žibert et al. [207] combined cepstral and phoneme recognition features to improve the accuracy of speech/non-speech segmentation. Several works also exist on speech/music discrimination, such as Scheirer and Slaney [165] or Kim et al. [80].

Significant work has been done on controlling user interfaces by pitch-based voice commands [185, 188]. A common control by pitch uses humming (producing a tone at the lips with the mouth closed, "*hmmmm*"), which has been evaluated by users as more convenient than whistling. The methods proposed in this chapter will therefore classify the extracted segments of an input audio signal in three categories: speech, humming, and silence (including other sounds, such as breathing).

Pruthi et al. [154] published a segmentation method that can distinguish between humming

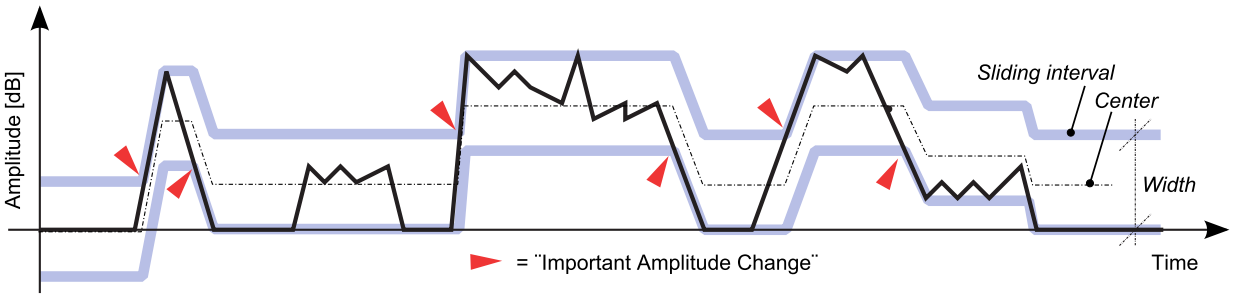


Figure 5.3: Segmentation using Energy Profile: Counting the important energy changes in a signal. This figure is taken from [182] with author's kind approval.

and other sounds, including speech. The method is based on computing features such as standard deviation of pitch, mean and standard deviation of a low-to-high energy ratio and the mean of the low frequency maximum. The limitations of this method are the need for constant pitch of the humming, since a maximum standard deviation of only 5 Hz is allowed. Another limitation is the fact that the input signal must continue for at least 400 ms before being classified as humming. Moreover, the method cannot perform speech segmentation. Almost no evaluation of the method has been described by the authors.

Neural network approaches have been used in several timbre recognition systems. For example, Hacihabiboglu and Canagarajah [54] used a multi-layer perceptron network for classifying short frames of musical instruments containing flute, clarinet and trumpet. Audio features were obtained from the discrete wavelet transform. The multi-layer perceptron was also used for classifying percussive sounds [195]. Several audio features were considered, e.g. zero-crossing, RMS, spectral centroid, or mel-frequency cepstrum coefficients. Another system developed by Fragoulis et al. [43] used an ARTMAP neural network to distinguish single notes played by five different instruments. Neural network approach has been also used in Vocal Joystick [14], in which the mouse cursor is controlled by vowels.

5.2.1 IAC method

The IAC (*important amplitude changes*) method [182] is an existing method for segmentation of humming and speech. It is based on an observation that the volume level of the sound changes more rapidly in a speech signal than in a non-speech signal (humming). In order to discriminate speech and humming, the method counts important amplitude changes in the energy profile of a sound signal.

An example how the amplitude changes are counted is shown in Figure 5.3. The method tracks the RMS energy level in the signal and adjusts the position $CENTER$ (dashed line) of a sliding interval of a fixed $WIDTH$ (solid blue line), so that the current RMS energy level (solid black line) is within this interval. The algorithm counts how many times the sliding of this interval changes its direction, i.e. how many times the RMS energy level starts exceeding the boundaries of the sliding interval one way or the other (red triangles).

The advantage of the IAC method is relatively simple and fast processing. The disadvantage is the requirement of certain amount of time in order to store enough frames to determine between speech and a non-speech sound. According to Sporka [182] this drawback can be overcome with careful design of the vocal gestures.

5.3 Segmentation method using MFCC and RMS

The segmentation method is based on classification of an audio signal by a neural network. The inputs of the network are features extracted from an audio signal. An audio signal recorded at 16 kHz is expected. First, the audio signal is divided into frames. Each frame contains 512 samples of the signal, and the step between two consecutive frames is 256 samples. The overlapping of the frames improves the time resolution of the method. Features are then extracted from each frame, as follows:

1. *Mel-frequency cepstral coefficients (MFCCs).* These coefficients are usually used in speech recognition systems [27]. First, the power spectrum of the signal is computed by the fast Fourier transform. Then the spectrum is mapped onto the mel scale [189] by a triangular band pass filter bank with 24 triangular filters. The MFCCs are computed by taking the discrete cosine transform of the logarithms of each band pass spectrum. The mel scale maps frequency to pitch, so that the subjective step in pitch is equal to the same step in the mel scale.

2. *Low and high frequency energy.* The energy was computed as the root mean square of the amplitudes of the signal after applying low-pass and high-pass filters. The cutoff frequency was set to 350 Hz, which was found as optimal. This finding is consistent with research of Pruthi et al. [154].

After extracting the features listed above from one frame of the audio signal, a vector of 26 features is obtained (24 MFCCs and 2 energy parameters). A multi-layer perceptron (MLP) neural network [161] is used for classifying the feature vectors. MLP is a feed-forward artificial neural network that uses supervised learning, and it is capable of approximating the outputs for a previously unseen input vector. The neurons in the MLP are organized into three layers. The first layer contains the input neurons. There are as many input neurons as there are features used for classification. The second layer is called the hidden layer, and we use 20 neurons in that layer. The last (output) layer has three neurons in our case. Each neuron corresponds to one class (speech/silence/humming). Before using the network, the weights of the neurons are trained by a back-propagation learning strategy. In order to achieve the best performance of a neural network, the number of training vectors for each class should be approximately equal. Therefore, the number of training vectors should be limited to satisfy this condition. Another reason for reducing size of the training vectors is memory limitation in MATLAB's Neural Network Toolbox. A simple linkage

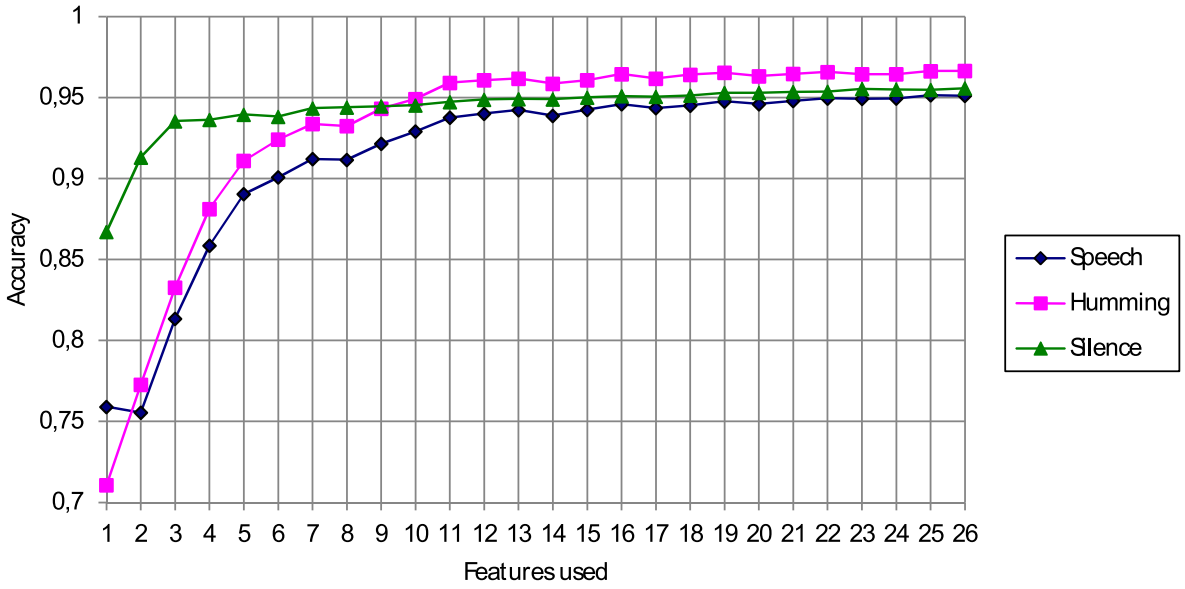


Figure 5.4: Classification accuracy of speech, humming and silence as a function of number of sorted features.

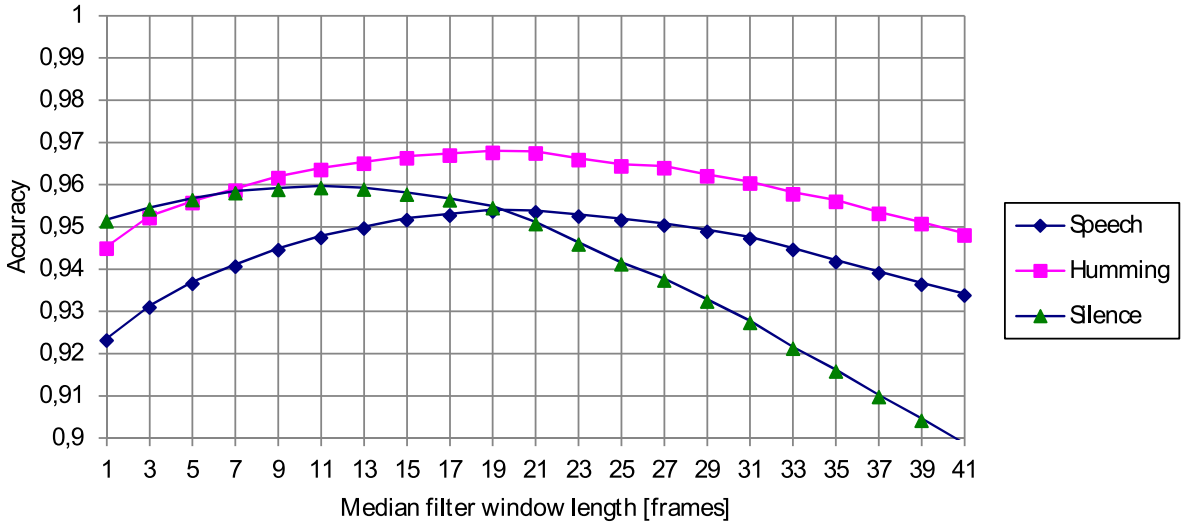


Figure 5.5: Classification accuracy of speech, humming, and silence as a function of length of median filter window.

clustering algorithm is used to create clusters of similar vectors. One representative vector is randomly chosen from each cluster, and these vectors are used to train the network.

As mentioned above, the total of 26 features are extracted from the audio signal. However, it is highly unlikely that each feature conveys information that is significant for classification. In order to select the features, we used the minimum-Redundancy-Maximum-Relevance (mRMR) feature selection method [148]. This method ranks features according to their mutual dissimilarity and their similarity to the classification. The features in the vector were sorted according to the rank obtained from the mRMR method. For the ranking purposes we used data described in Section 5.4.1. The effect on the accuracy of the method of selecting a subset of features is depicted in Figure 5.4. The figure shows that approximately the first twelve features convey significant information for classification. The use of more features does not significantly improve the classification accuracy. Reducing the set of features leads to a lower number of neurons in the input and hidden layers, and therefore to faster learning of the neural network. However, all 26 features were used for evaluation purposes.

The output of the MLP classifier is a sequence of frames labeled as speech, humming, and silence. A single frame of one class should never appear alone, surrounded by frames of another class, as the input audio signal contains segments of several frames of the same class. However, the classification method can misclassify some frames. For example, parts of words with nasal phonemes (m, n, η) can easily be misinterpreted as humming. To avoid such problems, a 1D median filter is used after frame classification. A single class of frame is always replaced by a dominant class within a window of N frames. The use of the median filter improves the accuracy of the segmentation method, but it introduces a delay of N/2 frames. The effect of window length on accuracy is depicted in Figure 5.5. For the purposes of the evaluation, the length of the window was set to 17 frames. The delay of the method was therefore 128 ms.

5.4 Evaluation

The method described above (further referred to as *MFCC method*) and the IAC method were subjects of an experiment. Both of them were used to find segments of speech and humming in a small corpus. The recognized segments were then compared to a gold standard (manually endpointed and labeled segments).

5.4.1 Experiment data

While no standard corpus of humming gestures and speech exists, the data had to be collected and annotated manually especially for the purpose of the experiment. The corpus was collected during a simulation of an interaction with a vector graphic editor controlled by spoken commands (such as "*draw line*", "*color green*", "*from here*", "*to here*") and

	Speech			Humming			Silence		
Speaker	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
1	0.99	0.99	0.94	0.99	0.99	0.97	0.98	0.97	0.99
2	0.99	0.98	0.98	0.99	0.99	0.98	0.98	0.97	0.98
3	0.98	0.97	0.96	0.99	0.99	0.99	0.98	0.96	0.97
4	0.99	0.99	0.97	0.99	1.00	0.97	0.98	0.96	1.00
5	0.99	0.99	0.98	0.99	1.00	0.98	0.99	0.98	0.99
6	0.99	0.98	0.97	0.99	0.99	0.97	0.98	0.98	0.99
7	0.99	0.99	0.97	0.99	1.00	0.96	0.98	0.96	0.99
8	0.98	0.93	0.95	0.98	0.96	0.98	0.97	0.97	0.96
9	0.99	0.97	0.97	0.99	0.99	0.99	0.98	0.98	0.98
10	0.99	0.96	0.97	0.99	1.00	0.98	0.98	0.98	0.99
11	0.98	0.97	0.97	0.97	0.98	0.92	0.95	0.89	0.97
12	0.98	0.94	0.98	0.98	0.97	0.99	0.97	0.98	0.94
13	0.98	0.99	0.94	0.98	0.97	0.98	0.97	0.95	0.98

Table 5.1: Speaker-dependent performance of the MFCC method. The neural network is trained separately for each speaker.

humming commands, which were similar to commands used in emulating the mouse cursor [185]. The whole utterances therefore matched the examples provided by Igarashi and Hughes [72].

A total of 25 minutes and 34 seconds of audio data was collected from 13 speakers (4 females and 9 males) at 16 kHz. The recordings were acquired in various conditions. Each speaker used a different microphone (headset, table microphone, or laptop built-in microphone), so the quality, background noise level, and volume level was different in each recording. The corpus contains 434 speech commands (each up to 4 words), and 579 humming commands. Speech utterances and humming commands were manually searched for in the audio data in order to generate a gold standard annotation. Each recording was randomly split in a ratio of 80:20, and 80% of each recording was considered as the training set. The rest was used for evaluating the two methods. The same split was used for each method. A total of 25 minutes and 34 seconds of audio data was collected from 13 speakers (4 females and 9 males) at 16 kHz. The recordings were acquired in various conditions. Each speaker used a different microphone (headset, table microphone, or laptop built-in microphone), so the quality, background noise level, and volume level was different in each recording. The corpus contains 434 speech commands (each up to 4 words), and 579 humming commands. Speech utterances and humming commands were manually searched for in the audio data in order to generate a gold standard annotation. Each recording was randomly split in a ratio of 80:20, and 80% of each recording was considered as the training set. The rest was used for evaluating the two methods. The same split was used for each method.

	Speech			Humming			Silence		
Speaker	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
1	0.96	0.90	0.92	0.96	0.91	0.95	0.97	0.99	0.96
2	0.98	0.97	0.93	0.98	0.95	1.00	0.96	0.96	0.94
3	0.94	0.85	0.89	0.96	0.93	0.98	0.94	0.96	0.85
4	0.97	0.99	0.84	0.98	0.95	1.00	0.97	0.95	0.98
5	0.97	1.00	0.85	0.98	0.95	1.00	0.98	0.97	0.98
6	0.97	0.99	0.85	0.99	0.95	0.99	0.97	0.97	0.99
7	0.97	0.98	0.92	0.98	0.93	1.00	0.96	0.97	0.94
8	0.92	0.82	0.80	0.94	0.95	0.84	0.94	0.91	0.98
9	0.94	0.89	0.82	0.96	0.92	0.94	0.97	0.96	0.98
10	0.96	0.94	0.77	0.97	0.94	0.97	0.98	0.97	0.99
11	0.93	0.94	0.84	0.96	0.96	0.92	0.93	0.85	0.96
12	0.96	1.00	0.81	0.98	0.99	0.97	0.95	0.89	0.99
13	0.92	0.76	0.95	0.93	0.96	0.82	0.95	0.96	0.93

Table 5.2: Speaker-independent performance of the MFCC method. The neural network is trained once for all speakers.

	Speech			Humming			Silence		
Speaker	Accuracy	Precision	Recall	Accuracy	Precision	Recall	Accuracy	Precision	Recall
1	0.93	0.94	0.68	0.91	0.75	0.99	0.94	0.97	0.92
2	0.81	0.73	0.40	0.76	0.60	0.95	0.89	0.97	0.74
3	0.68	0.39	0.80	0.71	0.85	0.45	0.92	0.91	0.85
4	0.91	0.85	0.59	0.89	0.75	1.00	0.93	0.98	0.87
5	0.89	0.77	0.56	0.83	0.74	0.92	0.90	0.92	0.82
6	0.95	0.80	0.95	0.95	0.83	0.96	0.92	0.98	0.89
7	0.77	0.55	0.66	0.78	0.61	0.68	0.90	0.99	0.79
8	0.86	0.67	0.57	0.78	0.61	0.83	0.81	0.87	0.71
9	0.91	0.88	0.67	0.90	0.78	0.96	0.95	0.97	0.93
10	0.95	0.82	0.85	0.91	0.76	1.00	0.90	1.00	0.83
11	0.89	0.80	0.88	0.89	0.78	0.91	0.89	0.95	0.71
12	0.93	0.87	0.82	0.92	0.83	0.96	0.93	0.96	0.86
13	0.81	0.60	0.70	0.82	0.74	0.71	0.91	0.92	0.86

Table 5.3: Speaker-dependent performance of the IAC method. Parameters of the method are computed separately for each speaker.

		MFCC speaker-dependent		MFCC speaker-independent		IAC speaker-dependent	
		Mean	SD	Mean	SD	Mean	SD
Speech	Accuracy	0.99	0.01	0.95	0.02	0.87	0.08
	Precision	0.97	0.02	0.92	0.08	0.74	0.15
	Recall	0.96	0.01	0.86	0.06	0.70	0.15
Humming	Accuracy	0.99	0.01	0.97	0.02	0.85	0.07
	Precision	0.99	0.01	0.94	0.02	0.74	0.08
	Recall	0.97	0.02	0.95	0.06	0.87	0.16
Silence	Accuracy	0.98	0.01	0.96	0.02	0.91	0.03
	Precision	0.96	0.02	0.95	0.04	0.95	0.04
	Recall	0.98	0.02	0.96	0.04	0.83	0.07

Table 5.4: Overall performance of the methods.

5.4.2 Results and Discussion

Accuracy, precision and recall measures are used for evaluating the two methods. The definitions of these values are shown in the following formulas. T_p stands for the number of frames with a true positive classification, F_p for false positive, T_n for true negative, and F_n for false negative.

$$accuracy = \frac{T_p + T_n}{T_p + F_p + F_n + T_n} \tag{5.1}$$

$$precision = \frac{T_p}{T_p + F_p} \tag{5.2}$$

$$recall = \frac{T_p}{T_p + F_n} \tag{5.3}$$

The aforementioned measures are usually used for the identification of two classes only. However, the same measures can be easily used for more classes as well. The values have to be then computed separately, i.e. each class is compared with the rest. When computing the accuracy for speech, one class contains only the speech frames, while the other class contains humming and silence frames. Precision and recall values are computed for the same two classes. The measures are computed similarly for humming and silence. This yields total of nine values expressing the performance of a method.

Results per speaker of both methods are shown in Tables 5.1, 5.2, and 5.3. Two variants of the MFCC method are evaluated: speaker-dependent and speaker-independent. In the speaker-dependent variant (see Table 5.1), the neural network is trained separately for each speaker. In the speaker-independent variant (see Table 5.2), the neural network is trained for all speakers together. Although the performance of the speaker-independent variant

slightly degrades, the overall values still outperforms the IAC method. As the parameters of IAC method (see Table 5.3) has to be adjusted per speaker, only speaker-dependent variant was taken into account. The average performances of methods and variants are summarized in Table 5.4.

Both segmentation methods can be compared from several points of view:

- *Robustness.* The MFCC method outperforms the IAC method in all measures, as shown in Table 5.4. The MFCC method is therefore more robust than the IAC method.
- *Method calibration.* The IAC method must be adapted separately for each speaker. However, the configuration process is simple and fast (in seconds). On the other hand, the MFCC method can be configured for several speakers at once (13 speakers in our experiment). The configuration process for this method takes a longer time (in minutes), as the neural network has to be properly trained.
- *Real-time application.* Both MFCC and IAC methods can be used in real-time application. The delay introduced by both methods is constant: 128 ms for the MFCC method, and 320 ms for the IAC method.

The results indicate that the both methods pose a promising step towards an assistive application that will provide interaction based on speech combined with humming commands. The short delay of the MFCC method is important for providing continuous control—while the speech segments are processed as a whole, the humming commands must be processed almost frame-by-frame to provide immediate feedback for the user.

5.5 Summary

A novel method for speech and humming segmentation is described in this chapter. The method is based on computing MFCC and RMS features which are processed by a neural network classifier. The method is compared to an existing one (IAC method) which is based on a simple observation that the volume level of the sound changes more rapidly in speech than in humming. Both methods were tested on a small corpus gathered from 13 speakers. The evaluation of the methods, detailed results, and discussion are placed in Section 5.4.2. As it was shown, the MFCC method outperforms the IAC method in terms of accuracy, precision and recall.

The first step towards developing an interactive assistive application operated by a combination of speech and humming is presented in this chapter. However, more work needs to be done. Formal descriptions of speech and humming exist separately, and they need to be combined to provide an easy-to-use tool for developers. There are also no design guidelines for an interaction that combines speech and humming. The guidelines will have to accrue from extensive testing with users.

6 Comparative Study of Pitch-Based Gestures

Number of vocal gestures has been used in order to control an NVVI application as found in related literature. These gestures have different requirements on the abilities of the user. For example, a short hummed tone is probably simpler to pronounce than a more complex melody. This chapter describes an experiment which aims on comparing basic vocal gestures in terms of perceived fatigue, satisfaction, and efficiency. The results of the experiment inspired a set of vocal gesture guidelines which may help in future vocal gesture designs. The research described in this chapter has already been published in [A1].

6.1 Motivation

Pitch-based input is the part of NVVI, in which the computer is controlled by the fundamental frequency of a sound signal. The user is supposed to produce a sound, from which the fundamental frequency can be extracted, for example, humming, whistling, or singing. Pitch-based input has been used as an input modality for people with motor disabilities [14, 185], a voice training tool [55], or mean for song query [46, 19, 226]. In these applications, short melodic and/or rhythmic patterns (referred to as *vocal gestures*) are used.

Previous research in this field mainly studied isolated instances of NVVI (such as mouse cursor control or computer games) and their performance. In most setups (see Chapter 2) the choice of the vocal gestures was made in a more or less ad hoc fashion. Questions of whether users prefer certain gestures over others, and why, have not been addressed so far in the literature.

This chapter presents an experiment with 36 participants. The goal of the experiment was to compare basic NVVI pitch-based gestures in terms of perceived fatigue, satisfaction, and efficiency. A paired comparison paradigm [194] was used. The results of the study inspired a set of NVVI gesture design guidelines that are presented at the end of the chapter.

The most common pitch-based gestures in current NVVI systems were selected for the experiment (see Section 6.3.3): flat tones, rising or falling tones, and a combination of rising and falling tones.

NVVI is a low-cost technique that is relatively easy to deploy, and may play an important role in the development of user interfaces for users with temporary disabilities (e.g. broken arms). While these conditions restrict the user's ability to use a keyboard and a mouse, investment in a more expensive assistive device would not be cost-effective due to the limited time for which the assistive device would be used. However, devices such as a mouse or a keyboard may be emulated by NVVI. This study was conducted within the framework of the VitalMind project [125], which focuses on the use of technology by elderly users who are considered to be one of the NVVI target groups as they are prone to temporary disabilities. For this reason, they were selected as participants in our study.

6.2 Related work

The applications of non-verbal vocal input can be roughly divided into two categories: real-time and non-real-time.

Real-time applications (continuous input channel) provide immediate feedback to the user while the sound is still being produced. This is useful, for example, for computer games [55, 184, 62] and interactive art installations [3]. NVVI thus does not work like speech recognition, where the system waits for the utterance to be completed. Another example is emulation of a computer mouse by pitch-based gestures [185] or timbre [14]. Both systems were evaluated in a study by Mahmud et al. [112].

Non-realtime applications (event input channel) of NVVI are applications where the user is expected to finish producing non-speech sounds before the system responds. Interaction with these systems follows the query–response paradigm, similar to speech-based systems. Applications of this kind are important for people who are not capable of achieving the level of speech articulation required by current automatic speech recognizers. Examples are emulation of computer keyboard [181], querying a song by humming [46, 212], and a command trigger [212].

NVVI shares some similarities with speech input (typically realized through automatic speech recognition, ASR). It utilizes the vocal tract of the user and a microphone that picks up the audio signal. However, the two interaction modalities are better fitted to different scenarios, so NVVI should be considered than a complement to speech input rather as a replacement for it. When comparing NVVI and speech input, several differences may be identified as listed in Chapter 2.

The performance of NVVI is usually lower than that of traditional input methods such as a mouse or a keyboard, but is still sufficient for cases when no alternative is available. For example, moving the mouse cursor using NVVI is about three times slower [112].

6.3 Experiment

The aim of this experiment was to rank the selected NVVI gestures by perceived fatigue, satisfaction, and efficiency, based on the participants' personal experience of producing these gestures.

The participants underwent a pre-test interview and a training period. They were asked to use the gestures in a test application in order to accomplish a series of tasks in a simple interactive scenario. Later, they were asked to perform *(i)* pairwise comparisons of the gesture sets and *(ii)* a comparison of individual gestures within each set, using a forced-choice questionnaire. They were asked which gesture seemed to them more tiring, more appealing, and yielding a quicker reaction from the system. Finally, insights were solicited from the participants in a post-test interview. All comparisons were within-subject.

We used the two-alternative forced choice experiment paradigm for ranking the gesture sets. This paradigm is commonly used in human-computer interaction research to obtain reliable subjective rankings of multiple objects or categories. For example, Čadík [20] used a similar setup to rank color-to-grayscale image conversion methods in a subjective study. Ledda et al. [90] employed the Law of Comparative Judgement (LJC) in a study of high-dynamic range imaging.

6.3.1 Organization

A total of $n = 36$ participants were recruited among students of the University of the 3rd Age. Each participant was asked to attend three sessions in the course of a single week. The data was collected after the last session. One session typically lasted half an hour. The participants received at least one day of rest between the sessions.

- *1^{st}session.* The purpose of the experiment was explained to the participant. A pre-test interview was conducted to learn more about the participant. The experimenter explained and demonstrated the function of the test application and the task that was prepared for the participant (see the next section for details). The participant was trained to produce simple gestures first (sets A and B in Figure 6.2) and then carried out the task (using each set twice). The participant qualified for the experiment after reaching 75% accuracy, which was typically after 15 minutes of training.

- *2^{nd}session.* The participant's ability to produce the gestures from sets A and B was verified. Then the participant was trained to produce the gestures in sets C and D. Then the participant performed the task twice, using each of sets A through D.

- *3^{rd}session.* The participant's ability to produce the gestures from sets A through D was verified. The participant was trained to produce the last gesture set, E. Then the participant performed the task twice, using each of sets A through E. The order of the gesture sets in this session was counterbalanced to control for learning effects. After all the tasks had been completed, the participant was asked to fill out the quantitative questionnaire and was interviewed and debriefed by the experimenter.

6.3.2 NVVI test application

A simple test application implementing an environment for synthetic GUI tasks was developed. The user interface of the application is depicted in Figure 6.1.

The task for the participants was to move the cursor (represented as a black ninepin) to the target (red and yellow circle) by producing the corresponding vocal gesture from the set that was being tested. This was repeated four times in each task. The direction of movement during a task was twice to the left and twice to the right. The positions of the

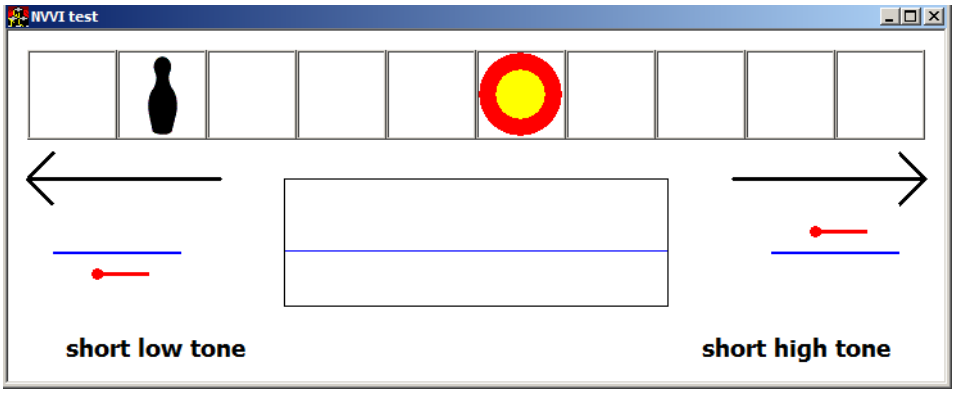


Figure 6.1: User interface of the application used in the experiment for movement along horizontal axis. The cursor is in the form of a ninepin.

target and the direction towards it were randomized in each run. The distance to travel was kept constant at 5 cells.[1]

The rectangle below shows the immediate feedback on the voice: the red line symbolizes the pitch of the tone and the blue line indicates the threshold pitch, separating the low and high tones. The threshold pitch can be adjusted to match the vocal range of each user. The vocal gestures to be used were depicted on the sides of the application window.

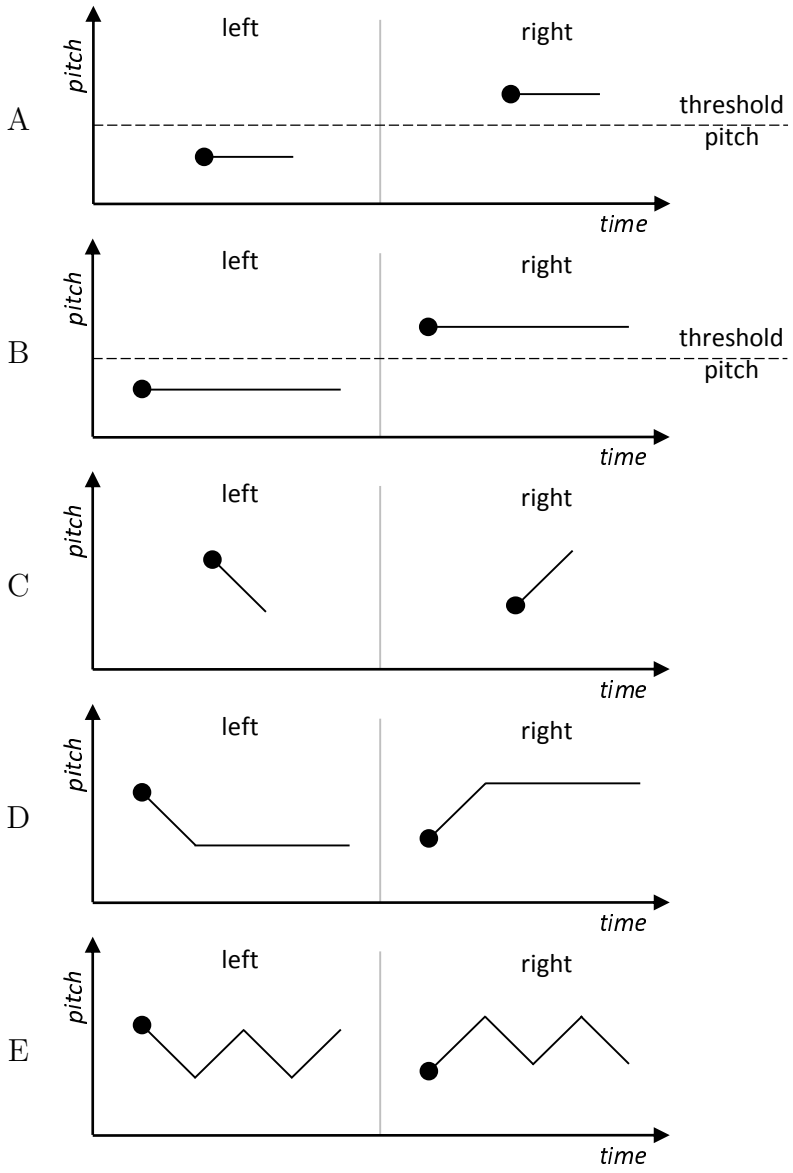
6.3.3 Selected Gestures

In this experiment, we used five different vocal gesture sets, as depicted in Figure 6.2. These gestures were commonly present in the current NVVI applications and research prototypes: Flat tones (differing by pitch, as in [181]), rising or falling tones (tones with increasing or decreasing pitch, as in [185]), and a combination of rising and falling tones (vibrato, as in [181]).

There were only two gestures in each gesture set. They were mapped to leftward and rightward movement. NVVI applications typically employ more than two gestures. The purpose of this setup, however, was not to test the simultaneous use of multiple gestures, but rather to expose the users to multiple gestures in a sequence, so that they could experience different kinds of gestures *in the same context of use* and thereby be able to compare them.

Both absolute-pitch and relative-pitch gestures were used, and also gestures employing a continuous input channel and an event input channel. An autocorrelation method [155] was used to detect the pitch of the sound. The method computes the fundamental frequency in a sound, so the participant could use any sound that contained this frequency. This includes humming *"hmmm"* as well as vowels *"a"*, *"ae"*, *"uw"*, *"ow"*, etc.

[1] A demonstration of the task performed using each of the gesture sets is shown in http://www.youtube.com/watch?v=LPoSIg7uNHY



Set	Description	Channel	Pitch
A	short flat tones	event	absolute
B	long flat tones	continuous	absolute
C	short inflected tones	event	relative
D	long inflected tones	continuous	relative
E	vibrato	event	relative

Figure 6.2: Gestures used in the experiment.

The gesture set A (Figure 6.2-A) contained short flat tones. The cursor was moved by one position after recognizing of the gesture (discrete, event-based control). Gesture set B (Figure 6.2) contained long flat tones. The cursor moved continuously until silence was detected (continuous control). The gestures in sets A and B differed in the pitch of the tone (the threshold pitch was calibrated at each session during training).

Gesture sets C and D (Figure 6.2-C and D) were similar to A and B, but a relative pitch approach was used. The movement of the cursor was determined by the initial tonal inflection of a gesture. A rising tone triggered movement to the right, while a falling tone triggered movement to the left ([184]).

Gestures E (Figure 6.2-E) were tones with oscillating pitch (*vibrato*). The first tonal inflection determined the movement of the cursor. With each following inflection, the cursor was moved by one cell (event-based input). The *vibrato* gestures were designed for rapid movement as long as the users could modulate their voice quickly.

6.3.4 Quantitative questionnaire.

The questionnaire was in two parts: *(i)* A pair-wise comparison of gesture sets and *(ii)* a pair-wise comparison of the two gestures within each set. A total of five gesture sets (A through E) were compared. The comparisons were based on the following forced-choice questions. The same questions were used for both *(i)* and *(ii)* with a slight difference of wording. The version of the questions for *(ii)* is marked by brackets []:

- (Q1) Which of these two sets of gestures [which of these two gestures] was more tiring for your vocal cords?
- (Q2) Which of these two sets of gestures [which of these two gestures] did you like more?
- (Q3) To which of these two sets of gestures [which of these two gestures] did the system react better?

Q1 was used as a definition of the physical difficulty of producing the gestures. Q2 and Q3 were aimed at satisfaction and efficiency, the usability attributes mentioned in ISO 9241-11 (via [67]).

For the 5 sets of gestures there would be 10 pair-wise comparisons for each question. In order to reduce the time burden, each participant performed only 5 randomly selected pair-wise comparisons for each question.

6.3.5 Participants

The participants (mean age=66, SD=5.9) were recruited by an advertisement in a local newspaper and from the University of the Third Age. There were 22 females and 14 males.

One fourth of the participants had an academic degree, and the others had a completed secondary education. The following information was collected in the pre-test questionnaire:

Health state. Five participants reported problems with their vocal cords, including hoarse voice and a mild form of dysphonia. One participant had difficulty in producing long tones, due to asthma. One participant had previously had a thyroid gland operation, which affected her performance. Three participants had a partial hearing loss; one wore a hearing aid.

Music experience Thirteen participants reported that they used to sing or play a musical instrument. Ten participants did not sing and had no music experience. Previous music experience was not observed to impact on performance in producing vocal gestures.

Computer experience. Eleven participants had a computer at home or at work, while three participants did not use computers at all. Some of them played logic games on their computers, such as cards, crosswords, Sudoku or chess.

6.4 Results

6.4.1 Quantitative results

(i) Comparison of the Gesture Sets: The first part of the questionnaire yielded a total of 180 pair-wise comparisons for each of the questions Q1, Q2, and Q3 from the total of 36 participants.

A frequency matrix of preferences was constructed for each question (see Table 6.1). We used Thurstone's Law of Comparative Judgments (Case V) [194] to obtain the interval $z-$score scales for the gestures. The $z-$scores are presented in Table 6.2.

(Q1) Set A (short flat tones) was the least tiring, closely followed by set B (long flat tones). The least favorable was set E (vibrato). (Q2) Set A followed by set B, was the most liked among the gestures. Set C (short inflected tones) and set E were the least favored in this aspect. (Q3) The best response from the system was reported by the participants when using set B. The worst response was reported when using set E.

(ii) Comparison of the Gestures within the Sets: The same group of participants also completed the second part of the questionnaire, in which gestures belonging to the same set were compared. Gesture set E was excluded from further data analysis as it was, in general, poorly accepted by the participants. 36 participants performed one pair-wise comparison per set of gestures (A through D) per question (Q1 through Q3). In each comparison, they could vote either for gesture *Left* or for gesture *Right*.

Table 6.1: Preference matrices for Q1, Q2, and Q3.

Q1	**Set A**	**Set B**	**Set C**	**Set D**	**Set E**
Set A	0.500	0.529	0.818	0.688	0.880
Set B	0.471	0.500	0.684	0.750	0.846
Set C	0.182	0.316	0.500	0.500	0.429
Set D	0.312	0.250	0.500	0.500	0.750
Set E	0.120	0.154	0.571	0.250	0.500

Q2	**Set A**	**Set B**	**Set C**	**Set D**	**Set E**
Set A	0.500	0.619	0.001	0.150	0.167
Set B	0.381	0.500	0.071	0.111	0.125
Set C	0.999	0.929	0.500	0.706	0.250
Set D	0.850	0.889	0.294	0.500	0.278
Set E	0.833	0.875	0.750	0.722	0.500

Q3	**Set A**	**Set B**	**Set C**	**Set D**	**Set E**
Set A	0.500	0.812	0.091	0.250	0.053
Set B	0.188	0.500	0.154	0.048	0.001
Set C	0.909	0.846	0.500	0.467	0.278
Set D	0.750	0.952	0.533	0.500	0.250
Set E	0.947	0.999	0.722	0.750	0.500

Probabilities of the column set being chosen over the row set.

Table 6.2: $z-$scores of gesture sets for questions Q1 through Q3.

	Set A	**Set** B	**Set** C	**Set** D	**Set** E
Q1	0.00 (5)	0.11 (4)	0.84 (2)	0.63 (3)	1.07 (1)
Q2	1.84 (1)	1.71 (2)	0.00 (5)	0.66 (3)	0.21 (4)
Q3	1.74 (2)	2.53 (1)	0.86 (3)	0.84 (4)	0.00 (5)

Note: The order is shown in brackets.

Table 6.3: Pair-wise comparisons between gestures of the same set.

	Set	**Votes for**		p-**value**	p-**value**
		L	**R**		**Bonferroni**
Q1	*A*: Short Flat	20	16	0.618	7.412
Q1	*B*: Long Flat	20	16	0.618	7.412
Q1	*C*: Short Inflected	10	26	0.0113	0.136
Q1	*D*: **Long Inflected**	9	**27**	0.00393	**0.0472**
Q2	*A*: Short Flat	16	20	0.618	7.412
Q2	*B*: Long Flat	17	19	0.868	10.415
Q2	*C*: Short Inflected	24	12	0.0652	0.782
Q2	*D*: Long Inflected	24	12	0.0652	0.782
Q3	*A*: Short Flat	19	17	0.868	10.415
Q3	*B*: Long Flat	19	17	0.868	10.415
Q3	*C*: **Short Inflected**	**30**	6	0.00006	**0.00072**
Q3	*D*: Long Inflected	26	10	0.0113	0.136

Legend: #L—number of votes in favor of gesture *Left* and #R—number of votes in favor of gesture *Right*. Significant differences are set in bold.

These comparisons could answer the following question: "*For one gesture set, is there a significant preference among the participants for one gesture over the other?*" This is a Bernoulli Experiment [146] for which a binomic test can be used. The null hypothesis holds that the true probability of either choice is 0.5.

Since a total of 12 comparisons was performed (3 questions $\times$ 4 sets of gestures), in order to reduce the risk of Type I error, a Bonferroni adjustment [168] of the $p-$value level was performed. In order for a result to be considered significant, the $p-$value must be less than 0.05/12. An overview of the results is shown in Table 6.3.

Gesture *Right* of set *D* (long inflected tones) was significantly more tiring (Q1) for the participants than gesture *Left*. A similar trend could be observed for set *C*, but the difference was not significant. For set *C*, the system was perceived to react significantly better to gesture *Left* than to gesture *Right*.

6.4.2 Qualitative results

Short flat tones. Participants did not have serious problems when producing short flat tones (set *A*). Two participants produced "*la la la*" instead of humming. This was not considered as an error, as the input was based on the pitch of the tone only. Several participants were confused about the direction of movement at the beginning of the task, and two participants had difficulties producing a correct tone, though they were able to complete the tasks successfully.

Long flat tones. The long flat tones (set B) task was also completed by all participants. They mostly appreciated the immediate feedback of movement and the simplicity of the gestures. They identified those gestures as easier and less fatiguing than other gestures, mainly because they did not need to repeat gesture by gesture and could do everything by one long tone.

Short tones with inflection. Most participants struggled with short tones with inflection (set C). Six participants were not able to learn these gestures at all, and therefore they could not complete the task. Approximately half of the rest had significant problems producing these gestures. Only one participant stated that these gestures were simpler than the others, because the absolute pitch of the tone was not important, and another participant enjoyed this task. Other comments were mainly negative. We observed that participants were more successful when producing the rising tone than when producing the falling tone.

Long tones with inflection. Participants faced similar problems with long tones with inflection (set D) to the problems with short ones. Again, falling tones were more difficult for some participants to produce than rising tones. Nine participants were not able to complete this task successfully.

Vibrato. The most difficult task was the vibrato (set E). Twelve participants skipped this task. They were usually confused by the direction of the gestures. Several participants identified these gestures as the worst. Only one participant liked the vibrato gestures more than short inflexion tones.

Participants differed in their comparisons of the long and short tones. Several participants claimed that long tones were more demanding than short ones, because they needed to hold their breath for a long time. On the other hand, several participants said that short tones were more demanding for them, because they needed to start the tone over and over.

The participants were asked to identify their favorite and least favorite gesture set. Seven participants liked flat tones, e.g. "*They were easier for me*", "*I did not feel embarrassed*". Nine participants disliked one of inflected tones (including vibrato), e.g. "*I do not have my voice trained enough*" The qualitative results suggest that flat tones (sets A and B) are more accepted than inflected tones (sets C, D and E).

Perception of Humming. Twelve participants did not feel comfortable. They mainly made comments such as "*I felt like a fool*", "*It was funny*", "*I felt like a small child*". Several participants also reported that the vocal gestures reminded them of animal sounds. However, five participants reported that they did not feel any embarrassment when producing humming.

Voice fatigue. Ten participants reported that they did not feel any fatigue during the experiment. Four participants complained about mild fatigue.

6.5 Discussion

The results presented above indicate that gestures using absolute pitch mapping (gesture sets *A* and *B*—flat tones) were well accepted by the users. Preference for a higher tone or for a lower tone were highly individual. These gestures can be used in both event and continuous input channels. The disadvantage of these gestures is the need for manual threshold pitch adjustment.

Gestures that use relative pitch mapping (gesture sets *C* and *D*—inflected tones) were found to be more difficult to produce, and were therefore not very well accepted by the users. An interesting point is that rising tones were significantly better accepted than falling tones. Only very few users accepted vibrato (gesture set *E*).

6.5.1 Guidelines for the Design of Pitch-Based Gestures

We have summarized the results into four guidelines for the use of designers of future pitch-based applications.

1. **Use flat tones if possible.** This experiment demonstrated that flat tones were easiest for the users to produce. This is consistent with the finding reported by Sporka et al. [181]. Any design of NVVI gesture assignment should therefore commence with flat tones. Other types of gestures should be used only in addition to flat tones.

2. **Use absolute pitch mapping if possible.** Absolute pitch mapping is better accepted by the users. Relative pitch should therefore be used only when more vocal gestures need to be assigned. In addition, splitting the vocal range into more than two vocal gestures is tricky, as more precise intonation is needed [186].

3. **Use positive rather than negative inflection gestures.** This experiment demonstrated that tones with decreasing pitch were more difficult to produce. If there is a need for relative pitch mapping, rising tones should be preferred over falling tones.

4. **Do not use more than one inflection per gesture.** Gestures with pitch oscillation were not well accepted by the users in this experiment. The existing literature reports successful use of gestures with a single inflection [184], but difficulties with complex gestures [181].

NVVI applications usually support more complex tasks than just one dimensional movement. An example of a complex task of this kind is playing the Tetris game [184] or

controlling a mouse cursor [185]. Designers may combine various types of gestures when a higher number of input signals are needed. Frequent operations should be assigned to simple gestures. For example, in Hands Free Mouse [A5] the short tone was used for the most frequent operation (left click) while scrolling was mapped to inflected tones.

6.6 Summary

The study described in this chapter has shown how users perceived various aspects of NVVI gestures: fatigue, satisfaction, and efficiency. Among numerous gestures that we could have chosen from, we focused on the basic pitch gestures that are present in the current NVVI literature: flat tones (i.e. tones with constant pitch), rising or falling tones, and gestures with oscillating pitch.

The study was performed with a group of 36 elderly users. Simple horizontal cursor motion tasks were used as stimuli for the participants. Each task could be carried out using only two gestures, for leftward motion or for rightward motion. The participants were exposed to five sets of gestures (10 gestures in total). They experienced different sets of gestures in the same context of use, and could therefore make a comparison between them. We used the paired comparison paradigm, which is commonly employed in the field of human-computer interaction for subjective ranking of stimuli.

The most acceptable sets were those with tones of constant pitch, followed by gestures with rising or falling pitch. Gestures with multiple changes of pitch (vibrato) were found unacceptable. Individual gestures were compared within the gesture sets. The users reported that a short rising tone was significantly less fatiguing than a falling tone.

A small number of design recommendations for pitch-based gestures were formulated. Any design of vocal gestures should start with flat tones, and other types of gestures should be included only when the required number of the gestures increases.

This study has focused on vocal gestures produced in a laboratory environment. A further study is needed to investigate the acceptability of NVVI in environments with a reduced amount of privacy: streets, offices, etc. In this study, NVVI was used by elderly Czech users. Levels of acceptance may vary in different social and cultural contexts. It will be interesting to study this aspect of NVVI in a cross-cultural experiment.

7 Scanning Keyboard Design

This chapter describes a contribution to the field of scanning text entry methods. Two new scanning techniques are presented: N-ary search scanning and a row-column scanning arranged on an array. Although a letter-level language model is used in both techniques, layouts of characters are static. In order to ensure the static layout, scanning sequence has to be dynamic. The effect of dynamic scanning procedure is studied in this chapter on six text entry methods.

The methods are operated by hissing—an interaction method suitable for severely motor-impaired people [3]. An experiment with 39 able-bodied users was conducted. The participants were able to enter a text by hissing at the speed of 2-3.2 words per minute. Experiment results have also shown that a scanning technique must be predictable for the users as they need to plan selections in advance. The research described in this chapter has already been published in [A7].

7.1 Motivation

Physically disabled people usually cannot achieve high entry rate due to their constraints. They have to use non-traditional input modalities which results in limited number of distinctive signals they can yield. Typical modalities may be a physical button, gaze interaction, intentional muscle contractions, or brain-computer interaction (see Section 3.5 for the full list).

A range of assistive techniques is available to help users with motor impairments. People, who are able to produce only one or two distinctive signals, can still use scanning technique for text input. As typing by scanning is very slow, various predictive techniques appeared. Predictive scanning keyboards often use dynamic layout - letters are rearranged while typing to achieve the optimal performance. However, such dynamic layout is rarely appreciated by the users as they always have to search for the desired letter. Effects of various scanning techniques and letter layouts on performance and user satisfaction are analyzed in this chapter. The expectation is that predictive keyboards with static layout will perform better than keyboards with dynamic layout.

One of the methods successfully used by people with special needs is the non-verbal vocal input (NVVI). The research of the NVVI can be roughly divided into three areas according to the main sound feature they are based on: pitch, timbre, and length. In pitch-based input the user can control computer applications by height of the tone. Whistling or humming are common sounds used in this interaction technique [212, 185, 181, 184, 188]. In timbre-based input, vowels are used to control various applications [14, 59, 116, 61, 60]. Length-based input has been so far used mainly for artistic installations controlled by length of hissing sound [1, 2, 3].

NVVI based on hissing has been successfully used by severely motor-impaired children [3].

To our best knowledge, no work exists on a text entry method operated just by hissing so far. Hissing is a method of interaction that can be used by paralyzed people or people with upper-limb motor impairments, such as quadriplegia induced by stroke, cerebral palsy, brain injury etc. Moreover, users are not required to have healthy vocal folds, as the sound of hissing is generated in the buccal cavity. However, they must control the breath to be able to produce hissing. The advantages of such interaction are language independence and fast and accurate recognition as opposed to speech [72]. Speech recognition software usually works relatively well for native speakers, however, the accuracy is much lower in the case of accented speakers or people with speech impairment.

7.2 Related work

Scanning is controlled by scan steps (i.e., highlighting items step by step) and scan selections (i.e., executing a command assigned to the highlighted item). Based on mapping of the user input on scan steps and selections, we can distinguish four scanning modes (see Section 3.2.2 for more details): automatic scanning, step scanning [127], self-paced scanning [41], and inverse scanning [209]. The text entry methods presented in this chapter use the automatic scanning as it is the most common mode. In this mode, the selection is controlled by the user input (switch activation) and step is triggered automatically after a scanning interval expires. Scanning is controlled by scan steps (i.e., highlighting items step by step) and scan selections (i.e., executing a command assigned to the highlighted item). Based on mapping of the user input on scan steps and selections, we can distinguish four scanning modes (see Section 3.2.2 for more details): automatic scanning, step scanning [127], self-paced scanning [41], and inverse scanning [209]. The text entry methods presented in this chapter use the automatic scanning as it is the most common mode. In this mode, the selection is controlled by the user input (switch activation) and step is triggered automatically after a scanning interval expires.

A scanning technique (see Section 3.2.2) describes grouping of items and how the scanning proceeds among them. Several scanning techniques can be distinguished: linear scanning [209], row-column scanning [83, 164], three-dimensional scanning [39] and binary scanning [64]. A model of a scanning systems in terms of a acyclic graph is called containment hierarchy [9].

As already mentioned in Section 3.3, text entry methods can be generally divided into two categories according to their layouts: *static* and *dynamic*. In static layouts (e.g., [31, 77, 65, 9, 12, 39]), the letters do not change the position. They are either displayed alphabetically or distributed in such a manner to gain maximum efficiency of the expert user of the method.

In dynamic layouts (e.g., [92, 126, 127, 159, 177, 142]), the position of letters is changed according to the actual probability in order to minimize the time needed to access a desired letter. Using the dynamic layouts increases cognitive demands on the user, hence they make

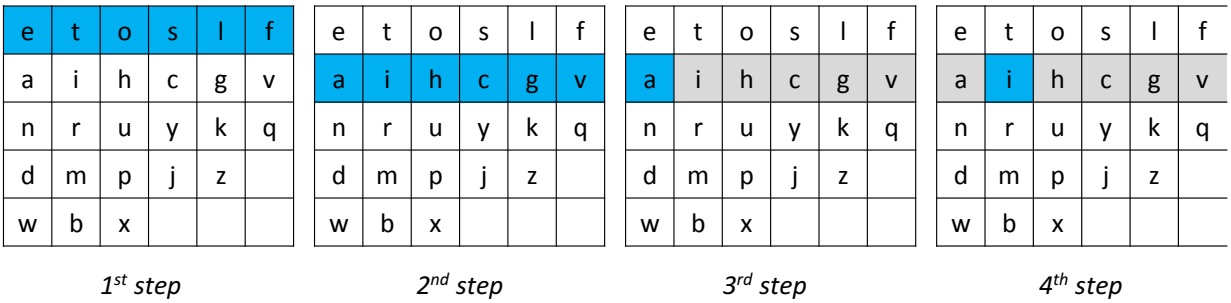


Figure 7.1: Row-column scanning. An example of selecting letter "*i*" by one row and one column step.

sense in special cases only. One such case is the scanning input. The letters are usually reorganized according to their probability in a given context. An exhaustive enumeration and evaluation of row-column scanning keyboards with static and dynamic layouts is given in work by Lesher et al. [92].

7.3 Scanning Techniques

This section describes scanning techniques used in the evaluation. A standard row-column scanning is described in detail and two new scanning techniques are presented: *N*-ary search scanning and a row-column scanning arranged on an array. In all techniques, a letter-level language model is used to adjust the scanning sequence according to probability of characters. See Section 3.4 for more details on language modeling. The adjustment is done in case of row-column scanning by rearranging characters in the matrix and thus using dynamic layouts of characters. In case of the two novel techniques, static layout is used, but the sequence of scanning is dynamically changed.

7.3.1 Row-Column Scanning

In the row-column scanning (e.g. [83, 164]), characters are organized in a matrix. Entering a character is done in two levels. In the first level, rows are sequentially scanned until a selection is made. Then, characters in the chosen row are scanned. When the selection is made, the highlighted character is entered and the scanning is reset to the first row. The process of entering letter "*i*" is depicted in Figure 7.1. It comprises of two scanning steps and two scanning selections.

In order to improve the entry rate of the row-column scanning, the characters in the matrix are rearranged according to its probability in a letter-level language model. The rearrangement is done with each character entered. Thus, the layout of characters of this method is dynamic. The selection sequence, however, is stable and follows a similar pattern. We can therefore say, that the selection sequence is static.

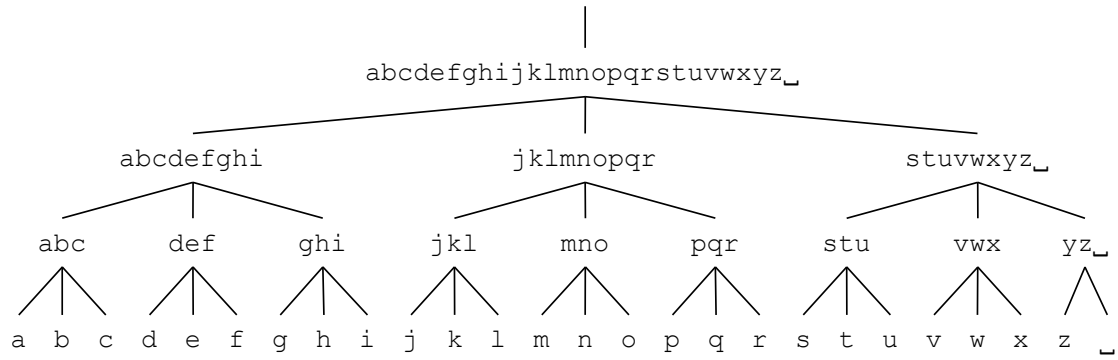


Figure 7.2: Containment hierarchy model for ternary scanning with lower-case character English alphabet with a space symbol.

7.3.2 N-ary Search Scanning

N-ary search scanning is a novel scanning technique. It has the advantage of static layout combined with letter-level prediction. The characters are always displayed in an alphabetic order. Such order simplifies the process of visual location of a desired character. In case of scanning keyboards with dynamic layout, users have to locate a character visually by linear scanning and they cannot rely on the visual memory. The locating process can be tedious for low-probable characters. On the other hand, in the static layout of alphabetically sorted characters the time needed to locate a character is then modeled by Hick-Hyman law [71] and is logarithmically dependent on the length of the alphabet. Locating characters visually in the static layout is hence faster than in the dynamic layout as logarithmic scanning is used instead of linear one. Moreover, users can rely on their visual memory.

With the alphabetical order of characters, the desired character can be entered by binary search algorithm adopted from basic programming techniques. The algorithm can be generalized to an N-ary search, where the alphabet is split into N groups. Scanning proceeds among these groups until the selection is confirmed. Then the group is split again and again until a desired character is found. Each character is located in the following number of levels:

$$levels = \lceil \log_N L \rceil \tag{7.1}$$

L is the size of the alphabet and N is the number of groups. Originally, the alphabet would be split into groups of the same size. This situation is shown in Figure 7.2 in terms of the containment hierarchy model. The scanning proceeds on each level until selection is made and then it descends to the lower level until a leaf is reached and the character is entered. The tree is balanced in terms of number of nodes and each character can be reached in the same amount of scan selection (27 characters results in 3 levels to scan per character).

However, the number of levels to scan can be improved by incorporating character probabilities. When the size of groups is balanced according to the probability of characters,

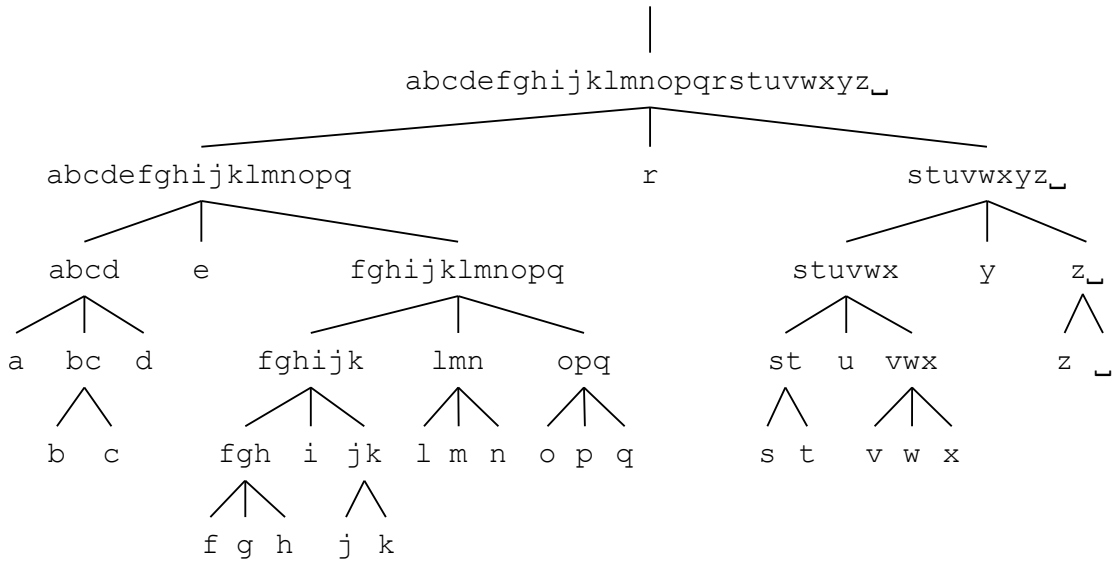


Figure 7.3: Containment hierarchy model for ternary scanning augmented with character probabilities. The context for this tree is "*The quick b*". Only lower-case character English alphabet with a space symbol are used in this example. Note that letter "r" has a high probability in the context.

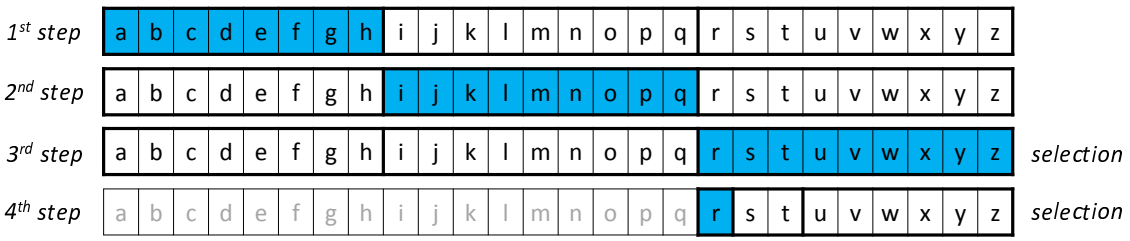


Figure 7.4: Typing "*r*" after "*Text ent*" on a ternary search scanning keyboard.

a character with high probability can be located in fewer levels. An example is depicted in Figure 7.3. It shows a containment hierarchy model for ternary scanning balanced by character probability after entering the context "*The quick b*". The phrase continues with letter "*r*", which is located on the second level and can be entered by only one scan selection. Note, that the user is inevitably slowed down entering low probable characters as more scan selections have to be triggered (e.g., letter "*f*" in the example).

In the modified N-ary search by the character probability, the alphabet is split into N groups with balanced probability in each level of scanning. In other words, the sum of probabilities of characters in each group is as close to $1/N$ as possible.

An example of user interaction with a keyboard that utilizes ternary search scanning is depicted in Figure 7.4. Let us assume that "*Text ent*" is the already entered text and "*r*" is the letter to enter. In the first level, the alphabet is split into three groups "*a–h*", "*i–q*", and "*r–z*". The scanning proceeds to the third group where selection is made. In

the second level, "r" is the only letter in the first group because of its high probability. Remaining characters are in the second and third group. The user chooses the first group and letter "r" is typed. In this case, the character was selected in two levels and four scanning steps only.

In each level, many possibilities exist how to split the alphabet and distribute the characters into groups. Exactly, the number of possible distributions M is given by Equation 7.2, where L is length of the alphabet and N is number of groups.

$$M = \binom{L}{N} \tag{7.2}$$

In order to find the optimal distribution, we define a matrix B (Equation 7.3) where indices of boundary characters are stored:

$$\mathbf{B} = [b_{i,j}]_{M \times N+1} \tag{7.3}$$

$$[b_{i \ldots M,1}] = 1 \tag{7.4}$$

$$[b_{i \ldots M,N+1}] = L \tag{7.5}$$

Each column of the matrix $\mathbf{B}$ is then filled with s unique permutation of the indices of boundaries. The first row contains index of the first character in the alphabet (Equation 7.4) and the last row index of the last character (Equation 7.5). Then we compute the vector $\mathbf{W}$ of weights for each permutation (Equation 7.7). The p_k is the probability of k^{th} character in the alphabet.

$$\mathbf{W} = [w_i]_M \tag{7.6}$$

$$w_i = \sum_{j=1}^{N} \left(\frac{1}{N} - \sum_{k=b_{i,j}}^{b_{i,j+1}} p_k \right)^2 \tag{7.7}$$

The optimal distribution is the one with minimal weight w_i. The corresponding boundaries are then in the i^{th} column of the matrix $\mathbf{B}$. Although this calculation has to be done in every level of the scanning, real-time operation can be easily achieved with the limited size of alphabet.

Note that unlike Huffman coding [70, 9, 158], in which characters are sorted according to their probability, our approach ensure the static layout of characters.

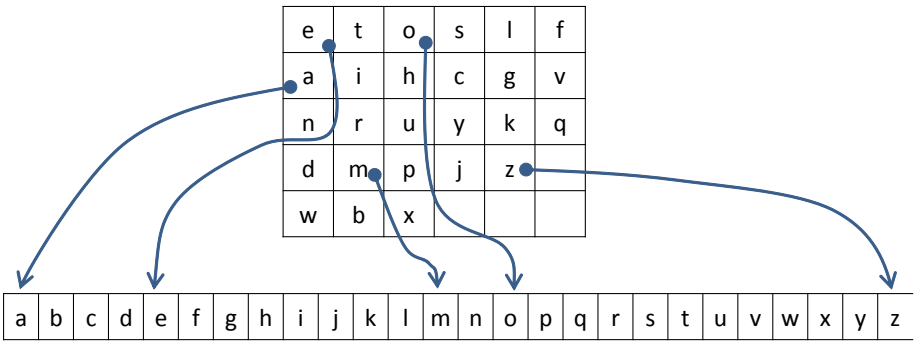


Figure 7.5: Mapping from matrix to array in row-column scanning.

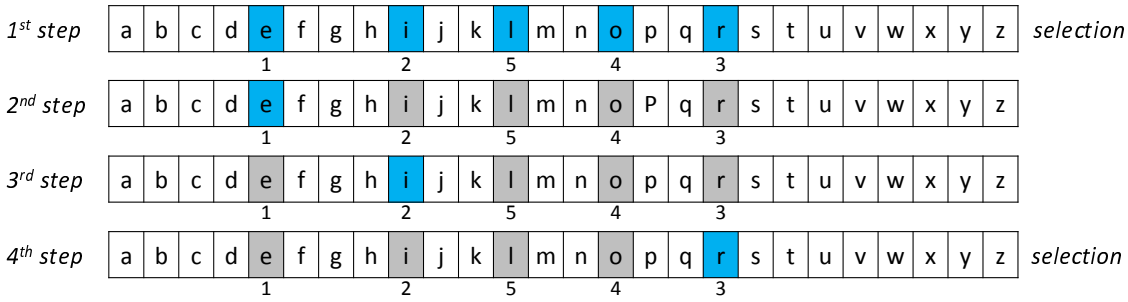


Figure 7.6: Typing "*r*" after "*Text ent*" on a keyboard with row-column scanning on an array.

7.3.3 Row-Column Scanning on an Array

Another novel scanning technique evaluated in this chapter is row-column scanning on an array. It is based on scattering the row-column scanning matrix in an array as depicted in Figure 7.5. This scatter ensures static alphabetical layout of characters even if dynamic layout has to be used in matrix due to the character prediction. Entering a character works technically in the same manner as in matrix row-column scanning hence a character is always entered in two levels. Theoretical performance is then the same, but the time required for visual search for a desired character is lower. Therefore, our expectation is that row-column scanning on an array will perform better than on matrix.

An example of user interaction with this scanning technique is depicted in Figure 7.6. Let us assume that the user has already entered the text "*Text ent*" and wants to continue by entering letter "*r*". In the first level, the user has to wait until the group that includes "*r*" is highlighted and make a selection. In our case, characters belonging to the group are those with blue background (Figure 7.6, 1$^{\text{st}}$step). In the second level, scanning is done only among characters of the group (gray characters in Figure 7.6, 1$^{\text{st}}$to 4$^{\text{th}}$ steps). The user has to wait until "*r*" turns blue (Figure 7.6, 4$^{\text{th}}$ step) and then make a selection.

Name	Character layout	Scanning sequence	Prediction	Scanning technique
Static RC keyboard	Static	Static	No	Row-column
Matrix keyboard	Dynamic	Static	Yes	Row-column
Array keyboard	Static	Dynamic	Yes	Row-column
Binary keyboard	Static	Dynamic	Yes	Binary search
Ternary keyboard	Static	Dynamic	Yes	Ternary search
Quaternary keyboard	Static	Dynamic	Yes	Quaternary search

Table 7.1: Keyboards used in the experiment.

0	1	3	6	10	15
2	4	7	11	16	20
5	8	12	17	21	24
9	13	18	22	25	
14	19	23	26		

letter	e	t	a	o	i	n	s	
freq.	.12	.09	.08	.08	.07	.07	.06	...
index	0	1	2	3	4	5	6	

Figure 7.7: Characters are assigned to cells of the scanning matrix ordered by probability. The cell with index "0" is the easiest to reach and the time needed to reach a cell increases with the index number.

7.4 Evaluation

In order to evaluate the different scanning techniques, a model of a scanning keyboard was built and its theoretical performance was measured in a simulation. Then, selected scanning keyboards were subject to a controlled experiment with users. After that, results stemmed from the simulation and the experiment were compared and the model was validated and updated accordingly.

7.4.1 Scanning keyboard designs

Six scanning keyboard designs were evaluated as shown in Table 7.1. As already mentioned before, scanning keyboards use either static or dynamic layouts. Generally, dynamic layouts are rarely appreciated by the users as the position of letters changes and users often have to search for a letter. The solution to this can be a static layout with dynamic scanning sequence.

Six scanning keyboards with different layouts were developed. Three of them followed the row-column scanning and the others used the N-ary search scanning techniques. The scanning keyboards are listed below:

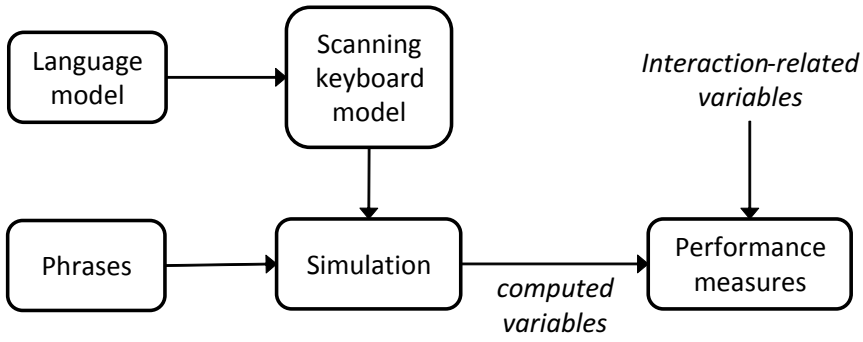


Figure 7.8: Analyzing scanning keyboard model by text input simulation.

Static RC scanning. The keyboard uses a static layout hence no prediction was involved. The letters are distributed according to their frequency in English. Figure 7.7 shows mapping of positions in the matrix letters. Numbers in the cells correspond to the indices of the sorted letter vector by frequency. For example, the letter "*e*", as the most probable letter, is located in the upper left corner and can be selected by only two short gestures with no scanning steps.

Dynamic RC scanning in matrix. The keyboard uses the same mapping of the letters to positions as SK1 keyboard, however, dynamic rearrangement according to the actual probabilities of the letters is used.

Dynamic RC scanning on an array. The keyboard uses the row-column scanning on an array as described in Section 7.3.3. The letters are always sorted alphabetically and displayed as an array.

Binary, Ternary and Quaternary search scanning. Three keyboards following the N-ary search scanning technique were subjects to evaluation.

The choice of interaction modality directly influences performance of the scanning keyboard. Users must be able to produce a signal fast enough to submit a selection before the scanning time runs out. Similarly, the initial scan delay must be longer because additional time is required to find the desired letter.

The scanning keyboards presented in this chapter use two hissing sounds: the short one to confirm selection and the long one for corrections. The threshold between the lengths of these two sounds is initially set to 500 ms, but it can be adjusted for each user. A hissing sound is recognized when root mean square (RMS) of the filtered input audio signal is higher than a threshold $TRMS$. This threshold can be also adjusted for each user.

7.4.2 Measuring performance

In order to measure the theoretical performance, a model of each scanning keyboard was build following the containment hierarchy concept [9]. The containment hierarchy was gradually updated from the letter-level language model according to current context.

The keyboard model was then matched to phrases in the simulation procedure (see Figure 7.8). Whole Mackenzie's phrase set for evaluating text entry techniques [106] was used as an input for the simulation. The simulation produced aggregated numbers of theoretical keyboard model performance, namely:

- C: Number of characters in the input text.
- S_i: Number of uninterrupted initial scan steps. Initial scan step delay is longer than a running scan step delay, therefore this variable was counted separately.
- S_s: Number of uninterrupted scan steps. All uninterrupted scan steps, except initial ones.
- G: Number of scan selections. All scan steps in which selection is made and hence the timeout is not reached. This number also corresponds to number of vocal gestures that has to be produced.

From these values, we can already compute some theoretical performance measures, namely *scan steps per character* (SPC), *gestures per character* (GPC), and *selections per scan step* (SPS). The SPC measure was introduced by MacKenzie and Felzer [104] for scanning ambiguous keyboard and is somewhat similar to keystrokes per character (KSPC) measure [99]. This measure includes all scan steps and selections. The SPC measure is in our case defined by Equation 7.8.

$$SPC = \frac{S_i + S_s + G}{C} \tag{7.8}$$

In order to express the number of hissing gestures per character, the GPC measure [216] was used. Gesture is regarded as an atomic operation—in our case a gesture corresponds to a single scan selection by the hissing gesture. The GPC measure is defined by Equation 7.9.

$$GPC = \frac{G}{C} \tag{7.9}$$

The SPS measure [104] captures the cognitive or motor demand on users. It is defined by Equation 7.10 With high SPS number the interaction is more demanding as the user does not have enough time to rest during the passive scan steps and to think over the selection strategy.

$$SPS = \frac{G}{S_i + S_s} \tag{7.10}$$

SPC, GPC, and SPS measures can be used for computing both theoretical estimates and empirical rates in an unchanged form. However, when estimating the resulting entry rate, we need to incorporate variables regarding timing, namely:

- D_i: Initial scan delay. The first scan delay after a letter is typed must be longer than other scan delays in order to give the user time to locate a desired character.
- D_s: Scan delay. A common scan delay is shorter than the initial one to improve performance of scanning.
- D_u: Time needed by the user to make a selection.

Using these variables, the entry rate in terms of words per minute WPM_{est} can be estimated by Equation 7.11.

$$WPM_{est} = \frac{C}{D_i S_i + D_s S_s + D_u G} \times 60 \times \frac{1}{5} \tag{7.11}$$

For computing the empirical WPM rate, we used a well-known formula (Equation 7.12) as described in the work by Wobbrock [216]. In this formula, T express time in seconds needed to transcribe $C - 1$ characters. SPC, GPC, and SPS measures

$$WPM = \frac{C - 1}{T} \times 60 \times \frac{1}{5} \tag{7.12}$$

Another variable that cannot be directly estimated in the simulation is the error rate (ER). The error rate (see Equation 7.13) was computed as a number of back actions divided by total number of user actions (i.e. back actions and selections). This formula resembles corrected error rate as defined by Soukoreff and MacKenzie [179] in unified error rate metric. However, the error rate defined in Equation 7.13 computes with selections and back actions rather than backspaces and entered characters, which makes it more appropriate for the scanning input.

$$ER = \frac{back}{back + G} \tag{7.13}$$

7.4.3 Experiment

The aim of the experiment was to analyze three research questions RQ1-RQ3 listed below.

RQ1 Predictive keyboards with static layout will perform better than keyboards with dynamic layout.

RQ2 The WPM measure (Equation 7.11) can be used for estimating performance of a scanning keyboard.

RQ3 Hissing can be used as an input modality for scanning keyboards

Participants. In the experiment, 39 participants (35 men, 4 women, mean age=21.3, SD=1.04) took part. They were recruited from university students. In order to avoid the carry over effect between N-ary search and row-column scanning keyboards, between-subject design was used. Two groups of participants were established, the first group evaluated row-column scanning keyboards (SK1, SK2, and SK3) only and the other one evaluated N-ary search scanning keyboards (SK4, SK5, SK6).

Organization. Each participant completed two 30-minute sessions. During the sessions they were asked to copy phrases taken from the Mackenzie's phrase set [106]. Participants were required to correct errors. In order to minimize the learning effect, the sequence of keyboards in each group was counterbalanced. After finishing both sessions the participants were asked to complete a post-test questionnaire to acquire subjective rating of each keyboard. The whole experiment was held remotely. The initial scan delay was set to 1500 ms and a scan delay to 750 ms for each user, which was identified empirically as sufficient for novice users in pre-studies.

7.5 Results

In the experiment, total of 399 transcribed phrases were collected in the first session and 515 in the second session varying from 76 to 121 phrases per keyboard.

Entry rate. The entry rate of each keyboard in terms of WPM rate is depicted in Figure 7.9. Results from both sessions as well as simulation results are shown. Using one-way ANOVA we found a statistically significant effect of the type of keyboard on entry rate ($F_{5,102} = 14.95, p < .001$). TukeyHSD test [170] was used in the post-hoc analysis for finding significantly different pairs. In the second session, the Matrix keyboard was found significantly faster than the other methods. Other differences were not statistically significant. Even though the theoretical speed of the Array and Matrix keyboard was the same, the measured speed was much lower. In case of the Array keyboard, it was found quite difficult to keep up with scanning by the participants. They often complained that the scanning technique is not predictable enough. Otherwise, the measured WPM values proportionally to values acquired from the simulation. They are, however, affected by misspellings and consequent corrections made in the real experiment by participants.

Gestures per character. The GPC measure express how many selections are needed to enter a character. The results are shown in Figure 7.10. The one-way ANOVA performed for data from the second session showed a significant effect of keyboard on the GPC rate ($F_{5,102} = 17.33, p < .001$). Binary keyboard requires significantly more gestures than all the other keyboards. Another significant difference was found between Array and Quaternary keyboards.

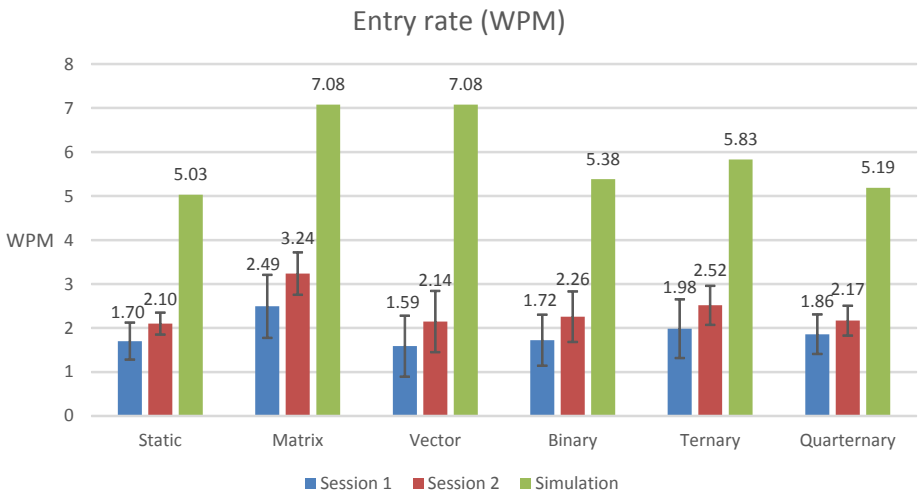


Figure 7.9: Average WPM rates for each keyboard measured in both sessions and in simulation. Error bars correspond to standard deviations.

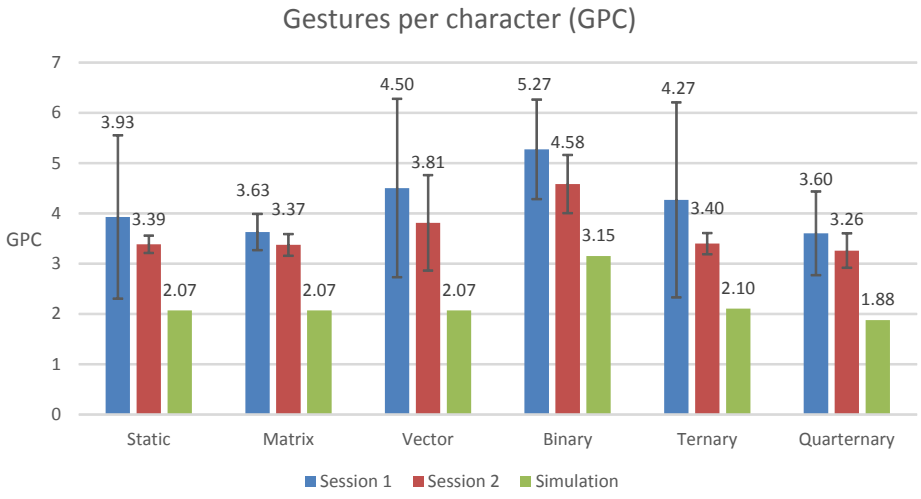


Figure 7.10: Average GPC rates for each keyboard measured in both sessions and in simulation. Error bars correspond to standard deviations.

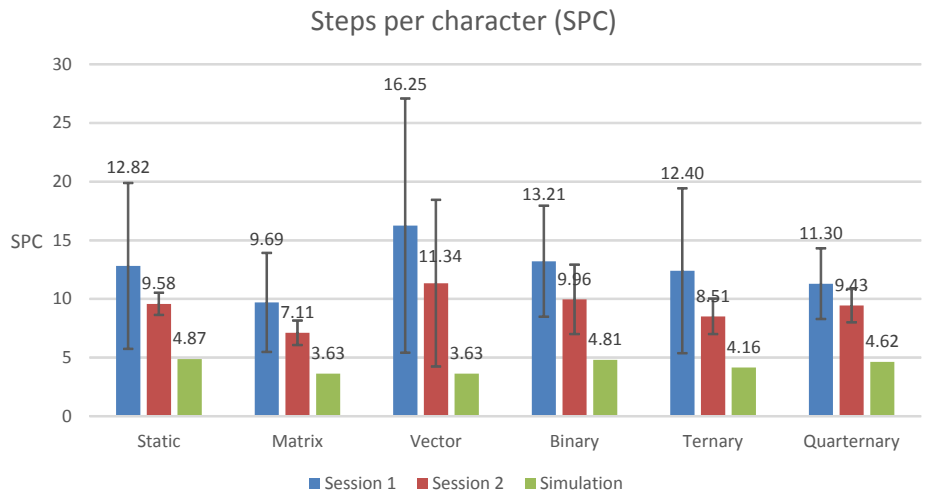


Figure 7.11: Average SPC rates for each keyboard measured in both sessions and in simulation. Error bars correspond to standard deviations.

Scan steps per character. The Figure 7.11 shows measured and simulated values of SPC rate. The one-way ANOVA performed for the second session showed a significant effect of the keyboard on the SPC rate ($F_{5,102} = 3.355, p < .01$). The only difference found in the post-hoc comparisons was between the Array and Matrix keyboards. The Matrix keyboard needs significantly less scanning steps than the Array keyboard. The poor SPC performance of the Array keyboard imply that users often missed the scanning step, in which selection should be made, and had to wait until highlighted once again. This of course slows down the typing.

Selections per scan step. The SPS values are shown in Figure 7.12. The one-way ANOVA performed for the second session showed a significant effect of the keyboard on the SPS rate ($F_{5,102} = 25.37, p < .001$). Post-hoc comparisons showed that Binary and Matrix keyboards have significantly higher SPS rate than the other keyboards. This results suggests that these two keyboards require higher motor demand on users [104].

Error rate. The error rate for each keyboard is shown in Figure 7.13. As the error rate cannot be estimated in the simulation, only values form experiment sessions are shown. A significant effect of the keyboard on the error rate ($F_{5,102} = 4.776, p < .001$) was found using the one-way ANOVA performed for the second session. Post-hoc comparisons showed that using Vector and Binary resulted in significantly worse error rate than using Static and Matrix keyboards. Other pairs were not significantly different.

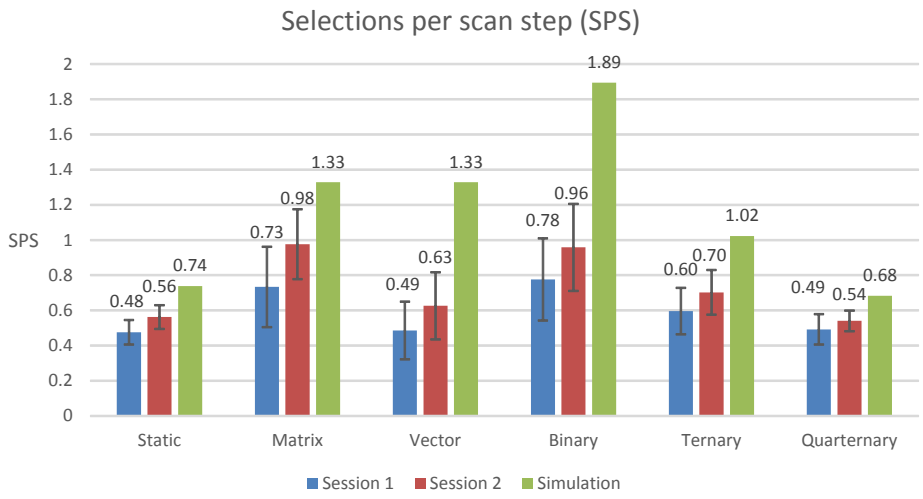


Figure 7.12: Average SPS rates for each keyboard measured in both sessions and in simulation. Error bars correspond to standard deviations.

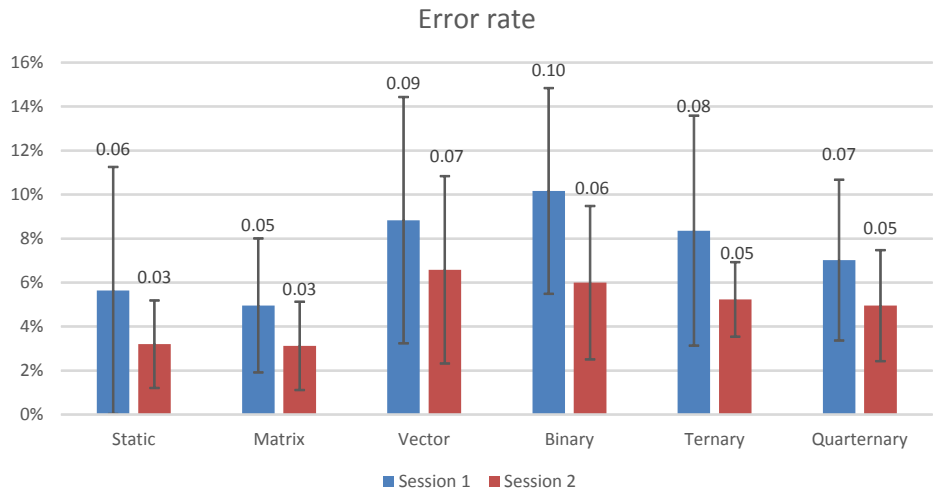


Figure 7.13: Error rate for each keyboard in both sessions. Error bars correspond to standard deviations.

Subjective results. In the post-test questionnaire, the participants were asked to evaluate each keyboard in terms of perceived accuracy, speed and comfort. Likert scale was used for each statement (e.g., "*The Static keyboard is accurate*"). The results are shown in Figure 7.14. In terms of accuracy, the most favorable keyboards were Static and Matrix keyboards, and the least favorable were Array and Binary keyboards. Regarding the perceived speed, the Matrix keyboard was rated as the fastest one, while the Array keyboard as the slowest one. The most comfortable was the Matrix, the least comfortable the Binary and Array keyboards.

The participants mostly complained about the Array keyboard. They mostly made comments about behavior of the keyboard such as "*It was like a lottery*", "*It was uncomfortable and hard to follow*". There were also several negative comments about the Binary keyboard: "*The scanning was too fast*". Most positive comments were made about the Matrix keyboard. One participant complained about fatigue and dry mouth after the experiment.

7.5.1 Discussion

In order to find the quality of the model as presented in Section 7.4.2, the WPM, GPC, SPC, and SPS results from the experiment were compared to the values estimated in the evaluation. Pearson's product-moment correlation [51] was applied to the mean values of the results of the six keyboards for the second session and simulation. No significant correlation was found for WPM ($r_4 = .58, p > .05$), and SPC ($r_4 = .18, p > .05$). A significant correlation was found for GPC ($r_4 = .94, p < .01$), and SPS ($r_4 = .81, p < .05$).

However, after plotting the data we could clearly see an outlier which affected the linearity of the data—the Vector keyboard. After removing the outlier from the data, the Pearson's product-moment correlation coefficient significantly improved. All results showed a significant correlation, namely WPM ($r_3 = 1.0, p < .001$), GPC ($r_3 = 1.0, p < .001$), SPC ($r_3 = .98, p < .01$), and SPS ($r_3 = .90, p < .05$).

Based on the aforementioned results of the simulation and the controlled experiment we can evaluate the research questions.

RQ1: Predictive keyboards with static layout will perform better than keyboards with dynamic layout. Our expectation that the N-ary and Array keyboards will perform better than the Matrix keyboard because of the static layout of letters turned out to be invalid. The objective results as well as participant comments lead to the conclusion that the correct choice of scanning technique is crucial for the scanning keyboard and surprisingly the layout of characters does not play such a significant role as expected. The scanning technique should be easily predictable by the user to allow planning of selections in advance.

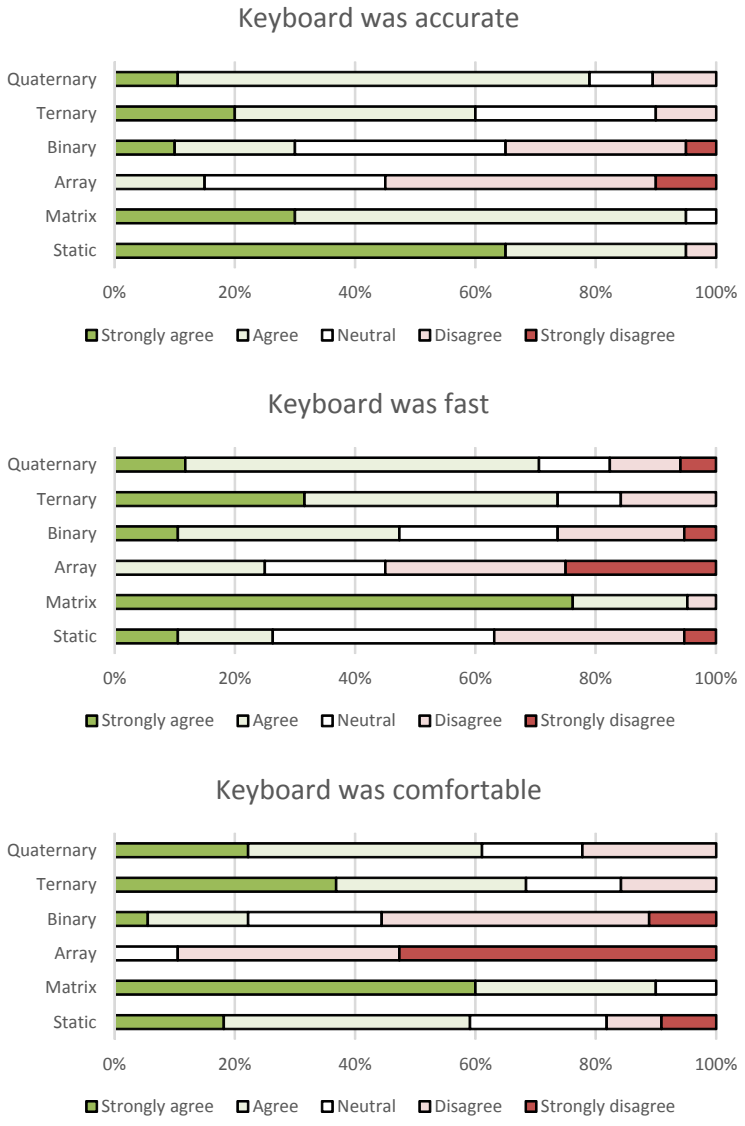


Figure 7.14: Subjective evaluation responses in terms of accuracy, speed and comfort.

RQ2: The WPM measure (Equation 7.11) can be used for estimating performance of a scanning keyboard. The WPM measure for scanning keyboards (see Equation 7.11) can be used for obtaining maximum possible speed that would be achieved by an expert user. The novice users reached approximately 40% of that speed after the second session. This, however, cannot be applied to the Array keyboard. The correlation coefficient between measured and simulated values was 1.0 after excluding the Array keyboard (.58 when included). Similar conclusion can be made for the SPC measure. On the other hand, GPC and SPS measures correlated for all keyboards as only selections were taken into account.

RQ3: Hissing can be used as an input modality for scanning keyboards. All participants were able to produce a hissing sound. Even though disabled participants did not take part in our experiment, we believe that hissing sound may be used by some of them as their impairment rarely affect the ability to produce hissing sound. This statement is supported by research of Al-Hashimi [3].

7.6 Summary

This chapter presents two novel scanning techniques the N-ary search scanning and row-column scanning on an array. They both ensure static layout of letters even though contextual probability of letters is used. The scanning techniques were evaluated in a simulation and a controlled experiment. The ternary search scanning keyboard was the best among N-ary keyboards in terms of speed and user satisfaction. However, it did not outperform the keyboard with dynamic row-column scanning in matrix.

The experiment resulted in an interesting conclusion that the layout of characters is not as much important as the scanning technique. Scanning techniques with an easily predictable sequence by the user are significantly better accepted than techniques that are not easily predictable. The users should not be forced to act immediately after single scan step as planning their actions in advance is important for them. We believe that this is the reason why array arrangement of row-column scanning failed. The N-ary search scanning, on the other hand, presented in this chapter is easily predictable and therefore accepted by the users, while keeping the static layout of characters.

8 Ambiguous Keyboard Design

This chapter introduces a text entry application for users with physical disabilities who cannot utilize a manual keyboard. The application allows the user to enter text hands-free, using the non-verbal vocal input. To keep the number of input sounds small, an ambiguous keyboard is used. As the user makes a sequence of sounds, each representing a subset of the alphabet, the program searches for matches in a dictionary. The ambiguous keyboard is based on a scanning ambiguous keyboard presented in work by Felzer et al. [34] and MacKenzie and Felzer [104]. The keyboard was redesigned and adapted to accept the alternative input signals. The usability of the software was investigated in an international longitudinal study done at locations in the Czech Republic, Germany, and the United States. Eight test users were recruited from the target community. The users differed in the level of speech impairment. Three users did not complete the study due to the severity of their impairment. By the end of the study, the users were able to enter text at rates between 2 and 3 words per minute.

The research described in this chapter has already been published in [A6]. The contribution of the author of the thesis to this research was redesign of the ambiguous keyboard, design of vocal gestures, implementation of the non-verbal vocal input, and conducting some of the experiment sessions in the Czech Republic.

8.1 Motivation

Interacting with a computer often requires entering text, especially when the computer is used as a communication aid. The standard PC keyboard seems perfect for this task since it provides a large number of keys. For example, experienced typist are able to enter text with a keyboard faster than they can utter the text aloud. However, it must be acknowledged that the standard keyboard is not perfect—if it were, it would be usable by everybody.

Unfortunately, persons with physical disabilities are often unable to operate a standard keyboard. Therefore, to fully utilize modern communication technology, alternative input methods are needed to enter text. Depending on the type and severity of the disability, the number of different input signals may be very limited. Sometimes everything has to be conveyed with a single touch of a button—the actuation of a single switch.

For users who can speak, automatic speech recognition (ASR) might be a faster alternative for text entry. However, the voice of many persons with physical disabilities is subject to dysarthria, so producing the same vocal output (with tolerable variations) for the same word is often impossible. As a consequence, ASR systems usually show poor accuracy [206]. The non-verbal voice interaction (NVVI), involving humming or whistling [212], is a possible answer.

This chapter presents the software system CHANTI, for "*voCally enHanced Ambiguous Non-standard Text Input*". The tool is based previously published scanning ambiguous keyboard QANTI [34, 104]. CHANTI tries to accelerate QANTI by using direct selection of keys by NVVI instead of scanning.

8.2 Related work

In ambiguous keyboards, multiple characters are assigned onto one key. Users can directly select a key to initiate a disambiguation process which results to entered text. Based on the kind of text entered we distinguish between letter-level and word-level disambiguation techniques. In the first technique, the text is entered character by character while in the other technique, whole words are entered. Examples from the mobile computing are MultiTap method (e.g., [147, 105] for the letter-level disambiguation and T9 method [52] for the word-level disambiguation. Please refer to Section 3.4.3 for more details on ambiguous keyboards.

The letter-level disambiguation has been used in the text input for motor-impaired people in the work by Mirro-Borras et al. [127, 126, 128, 129, 130]. However, the word-level disambiguation is more popular. It has been used, for example, in the work by Kushler [88] who describes an ambiguous keyboard with eight keys. Similar keyboards were described by Tanaka-Ishii et al. [191] and Harbusch and Kühn [65] which reduced the number of keys used to four.

In scanning ambiguous keyboards [104], the direct selection is replaced by scanning. Because of its efficiency with minimum input signals, scanning ambiguous keyboard became quite popular among text entry methods for motor-impaired people. For example, Kühn and Garbe [86] described a scanning ambiguous keyboard with four keys and Belatar and Poirier [12] presented a three-key keyboard on mobile device. Harbusch and Kühn [64] showed that scanning ambiguous keyboards outperform other scanning text entry methods.

8.3 Text Entry Method

As already mentioned above, CHANTI is built based on a scanning scanning ambiguous keyboard but accepts NVVI for directly selecting items. The next section reviews the design of the predecessor QANTI. Following this, the new approach is analyzed in detail.

8.3.1 QANTI, the predecessor of CHANTI

QANTI is an implementation of a scanning ambiguous keyboard (SAK), specifically targeted for persons with physical disabilities who are unable to utilize a standard keyboard. It was developed as a fully implemented system, rather than as a proof-of-concept prototype.

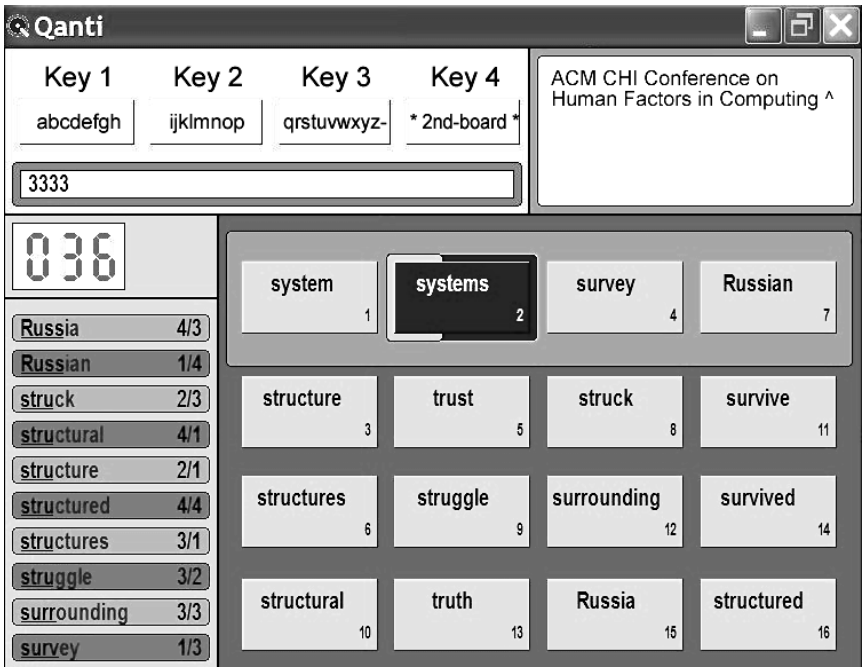


Figure 8.1: QANTI in candidate selection mode.

An exhaustive parameter search by MacKenzie and Felzer [104] identified a layout with four virtual keys as the most favorable (demanding the smallest number of scan steps per character for a given dictionary). Three virtual keys, each covering about one third of the alphabet, are used to produce a *code sequence*, while a fourth virtual key moves the focus to a frequency-ordered list of candidate words.

This is the basic concept of QANTI (see Figure 8.1 for a screenshot). To enter a word, the user first produces a code sequence with the help of the four linearly scanned virtual keys in the *sequence selection area* (top left of the screen). While the sequence is entered, a list of candidate words is constantly updated with the 16 most frequent candidates displayed both in the bottom left area of the screen (in alphabetical order) and on the buttons of a large 4×4 board on the bottom right (in frequency order).

Once the desired word appears, the user changes to *candidate selection mode*, where the 16 buttons are scanned in a *row-column* fashion (see Section 3.2.2 for further detail). Having selected the candidate, the user chooses among 16 finalization options shown on the buttons of the row-column scanning board. The options determine the way the selected candidate is rendered into the entered text (top right area of the screen), for example, turning the first character into a capital letter, or appending a space, comma, or period at the end.

For initiating selections, QANTI supports *intentional muscle contractions* [38] as an input signal. This feature emphasizes the target group, since it suffices to merely issue tiny contractions of a single muscle of choice, and thus requires a minimum of physical effort. As a consequence, even someone with a very severe disability can enter text reasonably

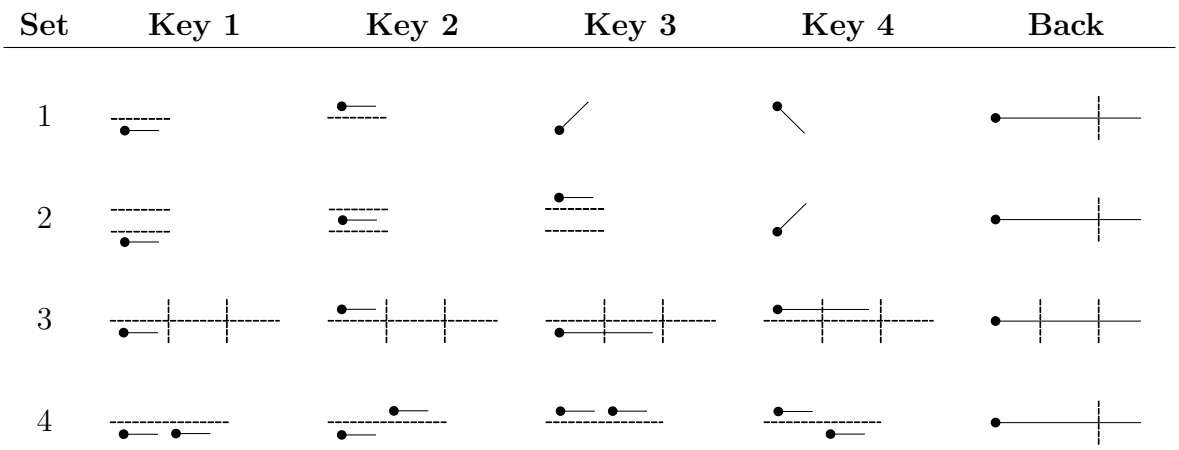


Figure 8.2: Gestures used for controlling CHANTI. Dashed lines represent thresholds.

fast, provided that one muscle (e.g., the brow muscle) can be reliably controlled (also [37]).

QANTI becomes a ready-to-use system through its menu mode. When the user applies a special mechanism involving the fourth virtual "sequence key", the buttons of the 4×4 board are re-labeled, giving access to several higher-order menu functions. In this mode, the user has the choice to correct errors (either in the entered text or in the current code sequence), to enter line breaks, to configure the scan delay, or to copy the entered text to the clipboard or to disk.

One menu option invokes an ordinary (non-ambiguous) on-screen scanning keyboard, offering a total of 64 virtual keys. This "Full Keyboard" allows the user to enter arbitrary character sequences, which is particularly helpful for entering non-dictionary words (e.g., names, passwords). This method adheres to a three-dimensional scanning technique [39] (with the 64 keys arranged in four groups with 16 buttons each, see Section 3.2.2 for more details). The menu also includes an add-to-dictionary feature for new words.

8.3.2 Design of CHANTI

CHANTI combines the philosophy of QANTI and non-verbal vocal input. The structure of the user interface is close to that of QANTI in that text is entered word-by-word. Words are selected from a vocabulary. Each word is ambiguously entered as a code sequence. After a code sequence is specified, the user disambiguates the selection by choosing the word from a list of candidates that correspond to the entered code sequence. Various functions, such as simple editing commands and character-based virtual keyboards for entry of out-of-vocabulary words are available through a menu.

As opposed to QANTI, where the interaction is performed by a single switch activated in specific time slots to make a selection, CHANTI is controlled exclusively by NVVI gestures. This provides faster access to the individual choices, compared to the scanning approach.

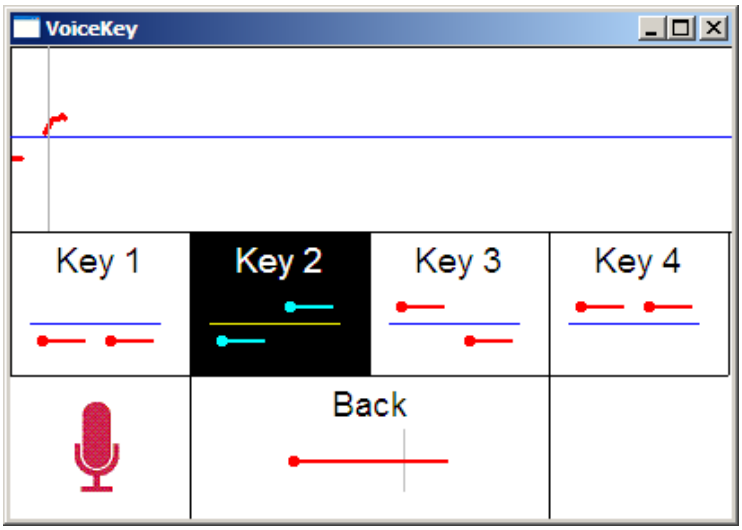


Figure 8.3: VoiceKey. The user has just produced a gesture G2. Upper part is the detected pitch profile. Lower part shows available gestures to produce. Vertical grey line is the length threshold.

An NVVI gesture is a tone or a sequence of tones with defined characteristics, such as pitch, pitch inflection, or duration, produced by the user. There were four different sets of gestures, equivalent in function, from which the user may choose.

There are four NVVI gestures in each set, *Key 1* through *Key 4* (used either to enter a code sequence or advance in the menu) and the *BACK* gesture to reverse the effect of the last gesture produced. This gesture can be used multiple times (multi-level undo). All gestures are shown in Figure 8.2.

NVVI has been implemented by a standalone program (see Figure 8.3) that recognizes gestures produced by the users and communicates them to the main application of CHANTI.

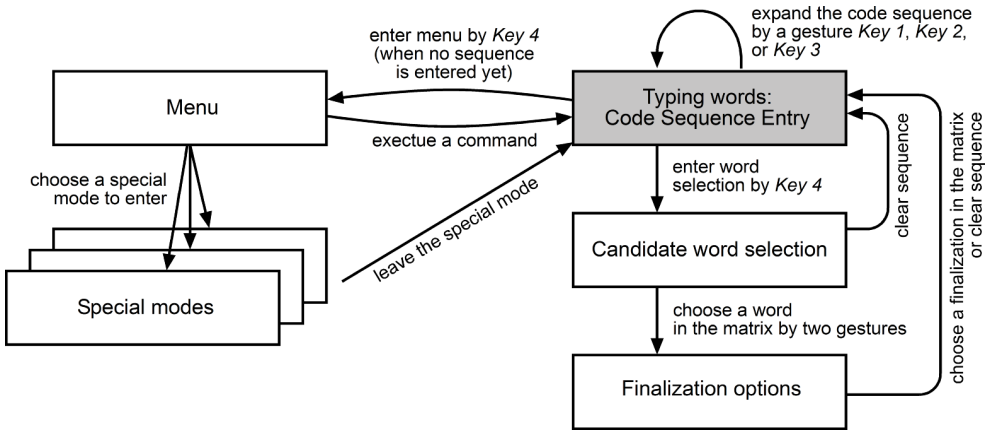


Figure 8.4: CHANTI user interface state diagram. The initial state is grey.

It also shows the continuous feedback of the tone (i.e., progression of pitch).

The structure of the user interface of CHANTI is shown in Figure 8.4. Initially, CHANTI awaits either one of the gestures *Key 1*, *Key 2* or *Key 3* to commence entering the code sequence (see a screenshot in Figure 8.5a) or *Key 4* to enter the menu. When a code sequence is entered, the gesture *Key 4* initiates candidate word selection mode (Figure 8.5b). The user then needs to produce two gestures: One selects a row, the other selects a column. Subsequently, the user chooses a finalization option (Figure 8.5c). The user may also choose to clear the code sequence and start over by selecting the "Delete Sequence" command in the candidate word selection or finalization option modes.

The menu allows the execution of simple commands (insert a space, remove the last character, remove the last word, insert a new-line character) and access to special modes. These are character-based virtual keyboards. The available characters on these keyboards are organized in a matrix. The user selects a desired character by specifying coordinates using the gestures (similar to selecting candidate words).

8.4 Evaluation

8.4.1 The Study Organization

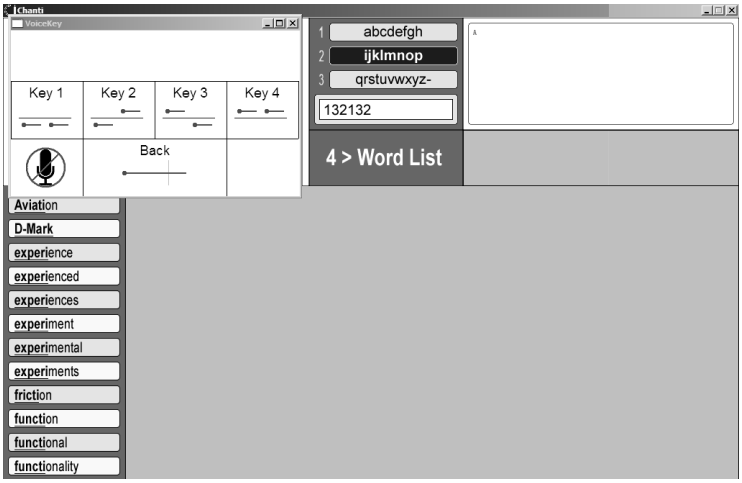
The purpose of the study was to gauge the first impression of CHANTI, how users would adapt to CHANTI over time, and whether they would be willing to accept CHANTI as their typing tool (and for which type of text). Since we aimed at studying users' insights, we organized the study as longitudinal and qualitative. Generally, participants were asked to use CHANTI for 30 minutes each day, over the course of 7 days.

For participation in the study we invited eight participants from three countries: Germany, the Czech Republic, and USA. This allowed us to test CHANTI in three different language contexts. The participants covered a range from no speech impairment to severe dysarthria. All participants were screened for being able to produce NVVI gestures during the Day 1. Three participants were excluded from the study because of the severity of their dysarthria which prevented them from producing the NVVI in required accuracy or extreme fatigue even after very short exposure to the system. In Germany, users were recruited via interviews with clients of several local healthcare institutions. In the Czech Republic, the users were recruited in cooperation with a local association of paraplegic people. In USA, the user was recruited through personal contact.

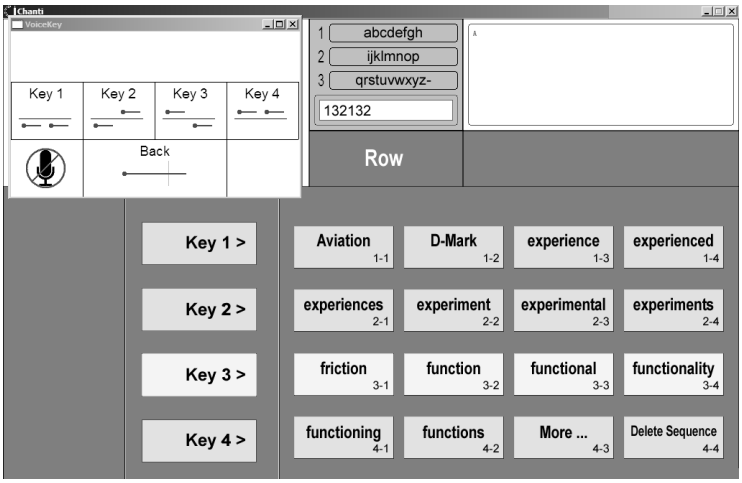
We used the same dictionaries for English and German as used in QANTI. The Czech dictionary was based on a frequency dictionary compiled at the Charles University in Prague [202].

The sessions used the following outline:

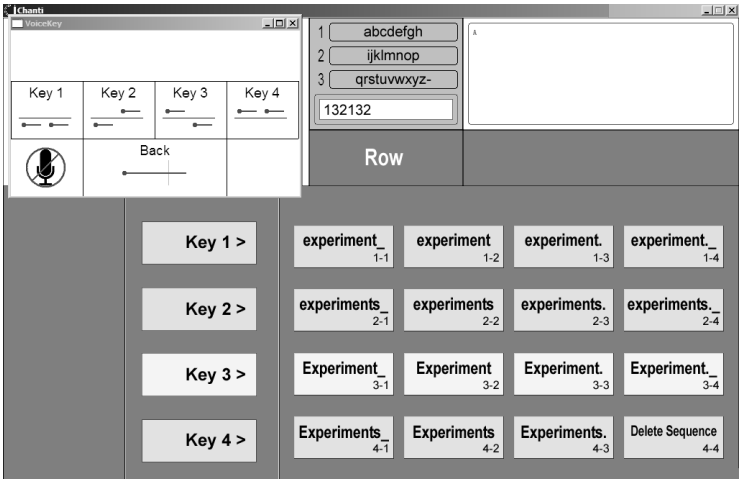
- Day 1: System set-up and pre-test interview: The participants were asked how they



(a) part of the code sequence entered



(b) selecting a word from the candidate list



(c) selecting how the word is finalized

Figure 8.5: CHANTI in various stages of operation.

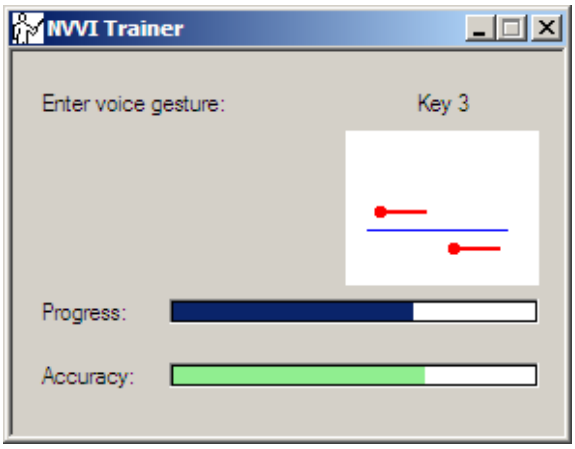


Figure 8.6: NVVI training module.

use the information and communication technology (ICT) as well as about their specific disability-induced problems relating to text input. The participants' assistive technology was discussed. The participants were trained in using NVVI (see below) and then determined which set of gestures was best for them.

- Day 2: The participants' capacity to use NVVI was checked. The session continued with the first exposure to CHANTI and first-impression interview.

- Days 3–6: Continued exposure to CHANTI: Participants were asked to begin writing using CHANTI. Their performance was measured on example phrases, between 5 and 15 words per phrase. These phrases were randomly selected from a text file. Separate collections of phrases were assembled for English, German, and Czech.

- Day 7: Last day of exposure to CHANTI. A post-test interview took place, in which the participants were asked about their overall experience using CHANTI, and what they considered the strong and weak points of the interaction.

8.4.2 NVVI Training

It is known that training is needed for proper use of the NVVI modality [112]. For this purpose, we developed a simple training module (see Figure 8.6) which generated a random sequence of gestures and then prompted the users to produce the gestures one-by-one. The users were considered ready for the study if they were able to produce 15 gestures out of 16 without a mistake.

8.4.3 Participant 1

Miloš[1] is 30 years old. He is quadriplegic since birth. He is an IT specialist in a small company based in Prague. On his request, his participation was remote. The interviews were conducted over the telephone and e-mail.

He is able to use a desktop computer while sitting at his desk by a mouth-held stick through which he can type on the keyboard as well as move the mouse. When laying in bed, his laptop is suspended on a platform above him, allowing him to use the stick to type on the keyboard as well as operate a small tablet that emulates mouse control. Apart from the Sticky Keys utility available in Microsoft Windows, he uses no other assistive technology. He reports typing as fast as 20 WPM. He spends 10 hours per day using a computer. He frequently uses shortcut keys. He likes exploring new technologies and interaction techniques.

Miloš is able to use the system tools of the machines he administers through a remote desktop facility. He frequently uses Microsoft Word, composes the HTML and PHP code, edits music, etc. He does not use social networks such as Facebook. He uses the ICQ instant messaging network for quick exchanges of short messages, rather than for extensive chatting. Regarding the use of computers, he feels no disadvantage against other users.

However, he relies on the help of others in hardware-related problems, including switching the machine on. He reports problems using the mouth-held stick on capacitive-sensing devices, such as touchpads or touch switches. He is willing to invest time in training new assistive devices and solutions. He reported having spent three months training for tablet use.

Interaction with CHANTI

Miloš was using CHANTI in Czech. Miloš was eager to participate in the test. He spent 12 hours using CHANTI over 7 days, which was well over the limit set by the design of the experiment. He spent about two hours testing all the sets and deciding which set of gestures to use. ("*I was trying hard to find out which set would be best for me*") Finally he opted for gesture set #1 (see Figure 8.2). He felt that these gestures were the easiest for him to produce and yielded the lowest error rate. He reported that gesture set #2 was the most difficult to use. ("*It was not easy to precisely hit those three separate tones*")

When using CHANTI to write unconstrained text, he noticed that the dictionary was not complete (the Czech dictionary contained only about 30,000 word forms) and tried composing words letter-by-letter, which he found very slow.

Initially, his median performance using CHANTI was about 1 WPM. The median performance over his later trials, measured using the internal test facility, was about 3 WPM (with in-vocabulary words only).

[1]The participants are represented by fictional names to protect their privacy.

He found the structure of CHANTI simple, yet versatile. He liked the fact that it can be controlled by only five distinct vocal gestures. He found CHANTI a useful tool for somebody who cannot use other means of control. However, he found typing using CHANTI too slow for his needs.

8.4.4 Participant 2

Petr is 19 years old. He is quadriplegic since an accident two years ago. He is a senior-year high-school student who spends typically 2 to 4 hours at a computer daily. He uses the computer to access study materials, to communicate with friends over e-mail, to access the telephone and watch movies. He relies on other family members to assist him daily with the hardware setup.

He uses head motion tracker SmartNav4 by NaturalPoint, which emulates a mouse and Click-N-Type software that emulates a keyboard. The device is based on tracking a reflective dot placed on the user's forehead. He is able to type at about 6 WPM using this setup. His performance is generally better when using a head rest or laying in bed. However, he reported that he cannot use glasses, as the reflection distorts the output from the device.

He reported limited experience with eye tracking technology, but had tried the system introduced by Fejtová et al. [32]. He was vaguely aware of breath controllers but he was not aware of other methods of the text input, including the use of the mouth-held stick.

Interaction with CHANTI

Petr was using CHANTI in Czech. Petr reported that he found the system easy to learn and that he was able to learn to use the system rather quickly. "*I only need to remember the ranges of letters for individual gestures, so that when producing gestures I would be too far off the range*"

Throughout the experiment, Petr used gesture set #4. He acquired a "steady rhythm" producing gestures, roughly at the rate of 50 to 60 gestures per minute which he would interrupt when making a decision, such as selecting a word from the list. In such moments, he would vocalize his thoughts in a soft voice, so as not exceed the volume threshold.

He used CHANTI in various locations with different acoustic qualities (kitchen, living room, office). He was visibly frustrated when acoustic interferences resulted in an undesired behavior of CHANTI. These interferences included background noises (he was not using a noise-cancelling headset) or a long reverberation of the room, causing the system to register longer tones than actually produced. However, he was aware of the need for calibration and requested it when he felt that it would improve the responses of the system.

He finds that the system would be best used for writing short text messages. ("*I would not want to write a whole novel using this*") However, he pointed out that since the Czech language uses an extensive system of declension and conjugation, a word often needed to

be specified until the very last letter, since only the ending of the word would determine the desired word form. This somewhat decreased the utility of the word prediction method used by CHANTI as opposed to English and its much simpler morphology.

His median typing rate was about 1.2 WPM on the first day of the data collection and about 2 WPM in the last day. He reported that in general he liked the input method but he felt frustrated when the word he was attempting to write was not in the dictionary and instead he had to type the word letter-by-letter using the virtual keyboard.

He expressed his wish to use the system again. He reported that he would use the system as an alternative to his current assistive technology when he gets tired moving his head. He and his caretaker were interested in a production-level implementation of the system.

8.4.5 Participant 3

Gabriele is 45 years old. Since her thirteenth birthday she has Friedreich Ataxia (FA). She used to work as an IT professional in university administration. She frequently wants to use a computer for education, entertainment, or gaming.

Due to the progressing symptoms of FA, she has significant problems using a keyboard. This keeps her from using a computer for communicating via chat or email. Typing one sentence can take up to fifteen minutes. Medium speech problems still make it impossible for her to use regular voice recognition. She immediately liked the idea of software that makes text entry more practicable for her.

Interaction with CHANTI Gabriele was using the German version of CHANTI. Gabriele decided that gesture set #1 was the easiest and most efficient for her. The symptoms of Friedreich Ataxia also heavily affect breathing and therefore speech and humming. She needed three hours before she was proficient enough with the training tool. The large amount of unintended input was frustrating for her: Not being able to switch the microphone off, she constantly produced input, for example, by coughing. It took her four days of intense practice with the experimenter before starting the first session.

Furthermore, she could hardly create humming sounds short and strong enough. The difficulty for Gabriele was to control her sound and breathing. While trying to make a short hum (*Key 1*, *Key 2* in gesture set #1), she could not control the timing. The hum has to be shorter than half a second, otherwise the program performs a *BACK* operation.

On the second day of the experiment, we prolonged the time threshold to 700ms. This meant that Gabriele was able to get *Key 1* and *Key 2* correct more often, but a longer time threshold also meant that she had to hum longer to perform a BACK key. A longer hum is more exhausting and also bears the danger of unintended inputs because of the difficulty to hold a tone for longer time (eventually the program would then rate a long hum as either *Key 3* or *Key 4*).

On average, she made 154 corrections for a sentence with 60 characters during the first

day of the experiment and did not exceed .5 WPM. She felt she would constantly improve. At the end of the experiment, the number of corrections dropped to 47 per 60 characters.

She only reached 1 WPM after the last session. This was mainly due to the number of corrections she was forced to make. In addition, her low typing rate was caused by a symptomatic eyes dysfunction, which complicates perception of information displayed on different parts of the screen. She would have preferred a bigger screen instead of the 14.1" notebook display used in the experiment. She indicated a willingness to continue practicing and working with the software on her own.

8.4.6 Participant 4

Rolf is 39 years old and has been diagnosed with Friedreich Ataxia at the age of 15. He uses a wheelchair since 1988, and he has considerable motor problems, which also affect his voice. Rolf is with a small software company and mostly works from home. His work requires him to use a computer for up to 8 hours every day, and despite the fact that his disease has progressed quite far, he is still able to use a standard keyboard and mouse (albeit at a modest typing rate of 3–6 WPM, depending on the time of day).

Due to his vocal difficulties, he is unable to use ordinary voice recognition software. He *is* able to communicate verbally, but the variations in his speech are too large for a computer program—he already tried several possibilities (mostly causing frustrating experiences). In addition, impaired fine motor control makes it difficult for him to use head/eye trackers.

Rolf is very motivated to find an assistive tool allowing him to interact with a computer at a rate comparable to an able-bodied person. When he was asked about being a participant in the CHANTI evaluation, he immediately accepted the invitation, gladly saying: "*This could help me a lot*".

Interaction with CHANTI

Rolf was using CHANTI in German. Rolf also decided for gesture set #1, even though producing ascending or descending gestures was not easy for him. Nevertheless, he indicated that this profile worked best for him, for example, as far as timing is concerned. When using CHANTI for the first time, the participant needed almost 16 minutes for a sentence with just 64 characters (entry rate: .8 WPM). The main reason for this was the heavy need for error correction (e.g., the *BACK* gesture) to take back erroneous selections.

However, the participant's results gradually improved. During the test week, Rolf spent more than 2 hours per day practicing with the program, and he was finally able to reach peak rates of 2.4 WPM. He reported that he liked the look of the program, and the colors. Besides, he commented: "*At the beginning, I spent a lot of time looking for the intended word. Later, I started to remember the position of the candidates, at least for frequent words; I'm sure I can beat my 'manual lower bound' of 3 WPM with longer practice*".

8.4.7 Participant 5

Sarah is a 32 years old graduate student. She is paraplegic due to a sporting accident that happened when she was 18. She was also diagnosed with thyroid problem, which made her suffer from a symptom similar to carpal tunnel syndrome last year.

Before last year, Sarah did not use any assistive system. However, because of her thyroid problem, since last year she had been using on and off speech recognition software and screen reader (to read out the information so that she can lay down while working without looking at the screen). She spends 6-10 hours a day with her computer on weekdays, although she usually does not use her computer on weekends and holidays. She uses various computer applications, including social networking (Skype occasionally), email (almost all the time), and Facebook (occasionally). She develops C programs as a part of her graduate work.

Interaction with CHANTI.

Sarah was using the English version of CHANTI. Sarah is a quick learner. She went from a median speed of .8 WPM on day one with a median correction of 16 out of 32 characters (50%) in her first session to a median of 2.5 WPM on the fourth day with a median of 5 corrections out of 27 characters (19%). It should be noted that on her fifth day, she was not feeling well, and while in the first trial on Day 5 she managed 3 WPM with only 3 corrections out of 28, her performance deteriorated quickly within minutes, possibly due to her thyroid problem, to 1.7 WPM and then down to 1.5 WPM with 16 corrections out of 24 characters.

We interviewed her at the end of her first day, and the day after her fifth session (she could not communicate effectively at the end of her fifth session). Her first impressions of the system were quite positive. She stated, "*After a while I was able to get more of the hang of it. I find it easiest, more convenient to use the Full Keyboard in CHANTI*". She did, however, complain about some of the key arrangements, stating that it was not intuitive for her that the shift key was in the bottom right quadrant of the full keyboard. She was using gesture set #1.

In her final debriefing, she made several remarks, which are summarized below:

- "Sometimes when using the word completion method of typing, the word I was looking for would never show up in the window. It was frustrating to have to delete each letter and then go to the full keyboard to type it up."
- "After a short while I felt that I was starting to memorize certain common inputs such as the space and I also felt I was becoming more proficient in writing, i.e., I was coming up with quicker ways of typing and looking for more efficient ways."
- "I also found it a little difficult with punctuation. I was hoping when using the word

completion interface that when typing '`I'm`' that I could type '`Im`' and when choosing the word that I meant that one of the options would be '`I'm`'."

When we asked her very specific questions, such as whether the system helped her to do her work, she provided many constructive criticisms that we can take home and use for further refinement of the system. The remarks are summarized below:

- On whether the system helped her to do her work, she said, "I found myself editing a lot of my typing and believe it could be made a bit more intuitive or perhaps have more functionality to help with short cuts."
- When asked about the on-screen real estate, she stated, "I would like to resize some of the keyboard windows so that I can see some screen space where I might be doing other work."
- On whether the system provided sufficient contextual help, she answered, "I found myself guessing as to what action to take to get the end result I wanted. Or sometimes I would choose some menu options believing that option would be available and when it wasn't I would go back."
- On menu arrangement, she thought it took a lot of steps to return to the main menu at times and sometimes she had to click on *DONE* or *BACK* to go back to the main menu which was not intuitive for her.

She did have several positive comments about the system. She thought that for most part the design was consistent, intuitive, and quite easy to learn, and she had fun playing with it. She said she would definitely use the system again on her own if she could get a copy of the system.

8.5 Discussion

Most users were satisfied with CHANTI and would use the method on their own if given the possibility. All participants who were able to use the NVVI notably improved their performance using CHANTI over the course of the experiment (see Table 8.1). The peak performance of the users was typically between 2 and 3 WPM. This is comparable to QANTI [34]. In fact, Rolf also participated in an earlier study evaluating QANTI—he reached around 2.4 WPM there as well.

Effects of the speech impairment. Though pitch-driven NVVI, as used by CHANTI, does not require the articulation of facial muscles as in speech interaction, the users still need to promptly control their breathing and vocal folds. In our study, we identified a "threshold of applicability" of NVVI: 3 participants out of 8 were not able to complete the first session (and therefore did not participate further). They had severe problems

Table 8.1: Overview of the results.

Participant (language and country)	Disability, condition	Dysarthria	Typical type rate (without CHANTI)	Mean type rate on first day*	Mean type rate on last day*	Gesture set	Would use CHANTI again?
Miloš (CZ)	Congenital malformation	No	20 WPM	2.8 WPM	4.2 WPM	#1	no
Petr (CZ)	Quadriplegia (accident)	No	6 WPM	1.2 WPM	2.4 WPM	#4	yes
Gabriele (DE)	Friedreich ataxia	Yes	very low	very low	1 WPM	#1	yes
Rolf (DE)	Friedreich ataxia	Yes	3–6 WPM**	.8 WPM	2.2 WPM	#1	yes
Sarah (US)	Paraplegia, (accident) thyroid problem	No	0–30 WPM	1 WPM	2.2 WPM***	#1***	yes

* Including the time spent on any corrections
** Depending on current condition
*** Sarah was using the *Full Keyboard* mode. Her peak performance was 3.2 WPM but during her last session the condition deteriorated and reached 1.6 WPM after an attack.

producing NVVI sounds due to their speech impairment induced by ataxia. The other two ataxic participants, Gabriele and Rolf, were able to produce NVVI sounds. While Rolf reached the performance similar to participants with no speech impairment, Gabriele frequently needed to correct malformed NVVI gestures, and this limited her performance at 1 WPM. Clearly, her speech impairment was more severe than that of Rolf. Gabriele was on the borderline of the target group. In particular, her speech problems caused a lot of frustration at the beginning (before she was able to proceed with the first test session), which almost made her decline participation in the experiment.

The NVVI threshold of applicability is lower than that of automatic speech recognition (ASR), as evident by Rolf's participation. Rolf reported that he could not use any system of speech recognition for interaction with the computer due to a different quality of his speech that was not compatible with the current ASR engines. Rolf represented "an ideal target group" of NVVI; i.e. the people who are able to speak but who cannot use the ASR as NVVI is more robust to speech impairments than the ASR. NVVI thus expands the range of applications of the vocal modality in assistive technologies by an important

margin.

The capacity to speak in a person can change notably over a short period of time and thus the performance of NVVI can vary, as documented by Sarah's participation on the last day: Sarah started out with typing rate of 3.2 WPM but during the session her condition worsened rapidly and her performance dropped to 1.6 WPM.

Language model. While NVVI is intrinsically language independent, CHANTI is by design intended to be used in the context of a specific language. When exploring the potential of CHANTI, Sarah and Miloš found the dictionary limited. Both users tried the Full Keyboard mode whereby any string could be typed. Miloš found the method very slow and would use it only to write a specific out-of-vocabulary word while Sarah switched to this method entirely. Her choice could be compared to switching off the predictive text entry method T9 available on mobile telephones. In Kurniawan's study [87] the participants were not using T9 as they found it distracting especially because of the incorrect predictions and subsequent recovery. This is supported by a study published in a work by Gutowitz [53]. According to his research, the T9 users mostly complain on missing words, no slang words or abbreviations, or inability to mix languages. Non-users of the T9 method mostly complain on its immediate usability (e.g., too difficult, or it suggests wrong words).

To fully accommodate the Czech language and its complex morphology, the method of the word selection should be changed. For example, the nouns could be selected in their basic form and the desired case could be chosen only in the next step. This would effectively reduce the size of the dictionary and thus the need for extensive browsing of the list of candidates.

8.6 Summary

When mentioning the acoustic modality for text entry, very often the automatic speech recognition (ASR) is mentioned as an example of the assistive technique suitable for this task. However, not all motor impaired people can use ASR due to the speech impairments that accompany their motor disability. This study has shown that the NVVI is a viable acoustic modality for text entry even for some of those who are not capable of speech intelligible by the ASR due to conditions such as ataxia.

This chapter presents CHANTI, a text entry method based on a combination of an existing scanning ambiguous keyboard QANTI and NVVI. The main goal of the presented study was to see how users learn using CHANTI during their initial exposure to the system.

CHANTI did not outperform some assistive text input techniques in terms of the type rate: By the end of the study, the users were able to enter text at typical rates between 2 and 3 words per minute.

However, CHANTI uses standard off-the-shelf hardware with no modifications needed for the system to run and therefore is inexpensive to deploy. As suggested by one participant,

CHANTI can be used as an alternative system for specific conditions, such as when taking rest from the current assistive tool.

For languages with complex morphology (which includes the Czech language), the prediction mechanism should be more informed by the grammar of the target language so that the user may more optimally perform the disambiguation of the candidates. Also, the finalization options should be language dependent. The function of adding custom words to the dictionary should be enabled in CHANTI.

In the design of the study, we were following the methodology described by Mahmud et al. [112] who reported that the performance of the use of NVVI reached a plateau by day 5. We did not detect a similar pattern in our data. A continuous study mapping the learning curve should be therefore carried out to determine the typing rate of this method in experienced users.

The focus of this study was on testing of the main principle of CHANTI. For this reason, some functions (such as adding new words to dictionary) were omitted for reasons of simplicity and should be implemented before CHANTI is made available for practical use.

9 Predictive Keyboard Designs

This chapter presents *Humsher*—a novel text entry method operated by the non-verbal vocal input, specifically humming and hissing sounds. The method utilizes an adaptive letter-level language model for the text prediction. Four different user interfaces are presented. Three of them use a dynamic layout, in which n-grams of characters are displayed according to their probability in the given context. One interface utilizes a static layout, in which the characters are displayed alphabetically and a modified binary search algorithm is used for an efficient selection of a character. The interfaces were compared and evaluated in a user study involving 17 able-bodied participants. Then, case studies with four disabled people were conducted in order to validate the potential of the method for motor-impaired users. During the case studies, the interfaces were iterated in order to improve the experience for the disabled people. Also combination of humming and hissing was tested with the target group to see the potentials of such input. The average speed of the fastest interface was 2.8 words per minute (WPM), while the fastest user reached 6 WPM. Disabled participants were able to type 2.8–4.4 WPM after seven sessions. The most successful user interface was further analyzed and the effect of language model on its performance was measured. A model of the interface was proposed and validated. The research described in this chapter has already been published in [A4] and [A8].

9.1 Motivation

Research in the field of the text entry methods has been widely documented for some time. The dominance of the QWERTY keyboard is obvious on personal computers. One of the reasons is the fact that learning a new layout is a tedious process that can take more than 100 hours [174]. 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 are willing to accept a longer learning time.

The maximum realistic text entry speed can be defined as a speed of an experienced typist using ten fingers on the QWERTY keyboard. The speed would be approximately 40–80 words per minute (WPM) for a professional typist [213]. With this speed achieved there is a little space for any enhancements like predictive completion or dynamic layouts as this would effectively slow down the type rate.

Physically disabled people usually cannot achieve such a high speed due to their constraints. Their communication with computers is rather limited only to several distinctive stimuli: small number of physical buttons, joystick, eye-tracking, features of the electroencephalographic (EEG) signal etc. This situation opens the space for a research of new entry methods which will take into account various limitations of the motor impaired users and will increase the entry speed.

Currently, a range of assistive tools is available to help the 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 can be used by people with special needs is the non-verbal vocal input (NVVI). It can be described as an interaction modality, in which sounds other than speech are produced, such as humming, whistling, hissing, or vowels. Whistling or humming are commonly used sounds in the pitch-based input. It is an input technique, in which the user can control the computer applications by height of the tone. The pitch-based NVVI has been successfully used in a mouse emulation [185], keyboard emulation [181] or real-time games [184]. The vowel-based NVVI has also been used to control various applications, for example, mouse emulation [14] or a movement of an robotic arm [69]. NVVI based on hissing was used for artistic installations [3] such as Expressmas Tree, sssSnake, or Blowtter. Blowtter is a voice controlled plotter and it has been successfully used by severely motor-impaired children.

Humsher, a virtual keyboard described in this chapter utilizes vocal gestures—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 [72]. Speech recognition software usually works relatively well for native speakers, however, the accuracy is much lower for accented speakers or for people with speech impairments [56].

9.2 Related work

A wide range of text entry methods exist which target the motor-impaired users. We can notice that 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.

One of the technique, which is heavily used to accelerate text entry methods for the motor-impaired users, is *prediction*. As already mentioned in Section 3.4, it is based on a language model, which provides the method with characters or words with probabilities based on currently written context. In predictive systems, most widely used is the letter-level statistical model. The language model contains letter-level n-grams (a sequence of n characters) together with their probabilities. The order of the model further refers to the longest n-gram contained in the model.

One of the first prediction systems was used in the Reactive Keyboard [26]. It predicted possible words according to the context that had been already written using an adaptive

dictionary-based language model. Predicted candidates could be selected by the mouse cursor. Expert users of a QWERTY keyboard would be slowed down, however, authors suggest that such prediction is useful for poor typists or people with limited movement of upper limbs. VITPI text entry method [15] offered unambiguous parts of words found in a dictionary.

In Dasher [211], letter-level prediction was used to alter size of virtual keys. The characters are selected by moving the mouse cursor around the screen yielding one continuous gesture. The writing speed achieved with a mouse is approximately 20 WPM with experienced users reaching up to 34 WPM. 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 [199] found that an average writing speed of 17 WPM after ten sessions could be achieved. This speed was a large increase from the initial speed of just 2.5 WPM. Speech Dasher [205] 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 40 WPM.

GazeTalk [76] 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 a dwell-time selection system [75]. The average typing rate achieved by novice users was 3.2 WPM. Urbina and Huckauf [201] used prediction in text entry by eye gaze in hierarchical pie menus. Their approach reached 10–17 WPM.

Prediction is often used in scanning keyboards, for example, in Huffman scanning [159, 158], linear scanning [66, 142] or row-column scanning [77, 209]. In ambiguous keyboard design, prediction has been used in work by Mirro-Borras et al. [127, 126, 128, 129, 130]. Prediction can be also used to improve accuracy of a pointer [111] or to replace surrounding characters of a virtual QWERTY keyboard with more probable alternatives [123].

9.3 Methods

In this section, four interfaces of our predictive virtual keyboard, *Humsher*[1], are described. The interfaces are operated exclusively by humming, three interfaces have a dynamic layout, in which letters are rearranged while typing, and one interface utilizes a static layout.

9.3.1 Dynamic Layouts

The interfaces described in this section employ a dynamic layout. The n-grams, which are extracted from the language model, are offered sorted according to their probability. The probability is predetermined by already written text. Practically it means that after

[1] A demonstration video of the method is shown in http://www.youtube.com/watch?v=3qgKwkmpyeY

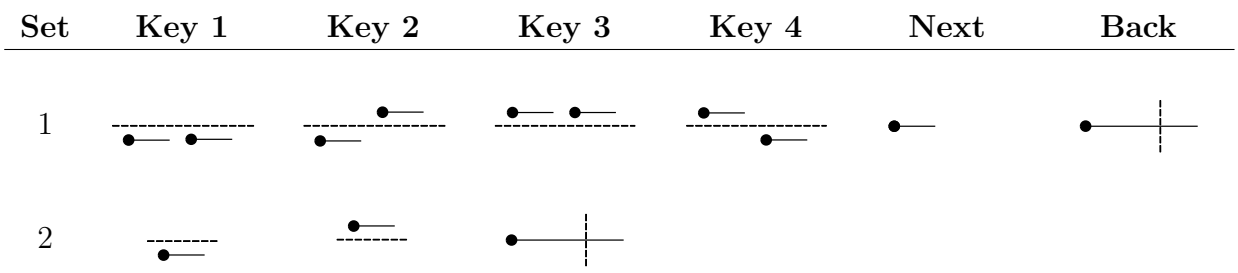


Figure 9.1: Gestures used for controlling Humsher. Dashed lines represent thresholds. Set #1 is used in Direct and Matrix interfaces. Set #2 is used in List and Binary interfaces.

typing an n-gram, the context is updated, probabilities of following n-grams are recounted and the layout is displayed accordingly.

Three different user interfaces (*Direct*, *Matrix* and *List*) with dynamic layout of characters were designed and implemented. Each interface differs in either vocal gesture set or in mapping of gestures to actions. The Direct and Matrix interfaces utilize six vocal gestures (see Figure 9.1, set #1) while the List interface utilizes only three simple vocal gestures (see Figure 9.1, set #2). The vocal gestures are explicitly identified by length (short/long) or by pitch (low/high). In order to distinguish low and high tones, a threshold pitch needs to be adjusted for each user. Only two different pitches were chosen as with increasing number of pitches, more precise intonation is required and the interaction becomes more error prone [183].

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 using the language model 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.

Direct interface

The *Direct* interface (see Figure 9.2) 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 Figure 9.1, set #1:

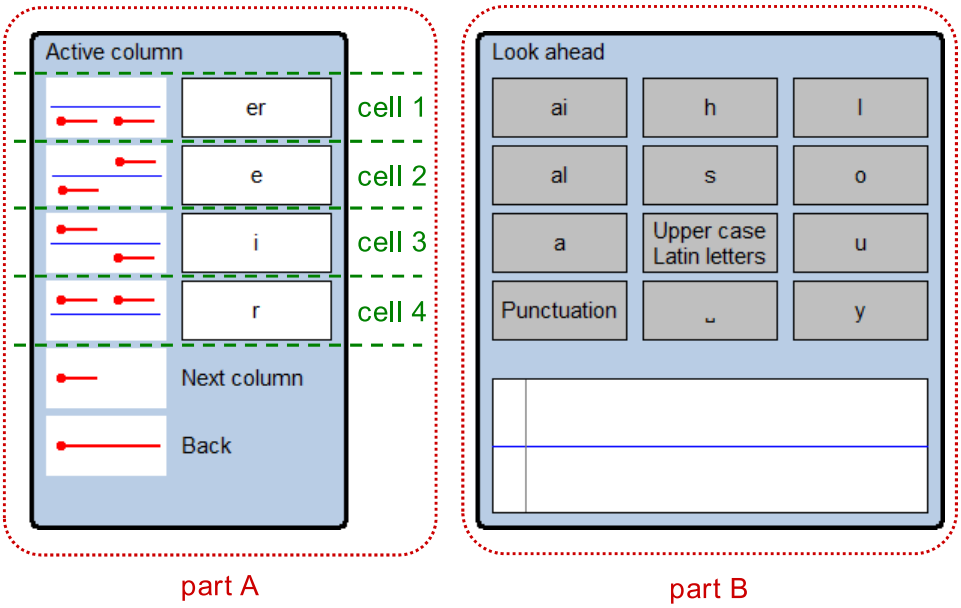


Figure 9.2: Direct interface. A—active column, B—look ahead matrix.

- *Key 1* two consequent low tones (cell 1),
- *Key 2* a low tone followed by a high tone (cell 2),
- *Key 3* a high tone followed by a low tone (cell 3),
- *Key 4* 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 Figure 9.1, set #1, *Next*) 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 Figure 9.1, set #1, *Back*). The longer the user keeps producing the tone the faster are the characters erased.

Matrix interface

The *Matrix* interface (see Figure 9.3) 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 while the rightmost cells contain the lowest probable n-grams. Selection of the correct cell is accomplished in two steps by direct selection of the desired column and row. First, the user selects a column by producing a corresponding vocal gesture (Figure 9.1, set #1, *Key 1–4*). The column is then highlighted and the same vocal gestures can be

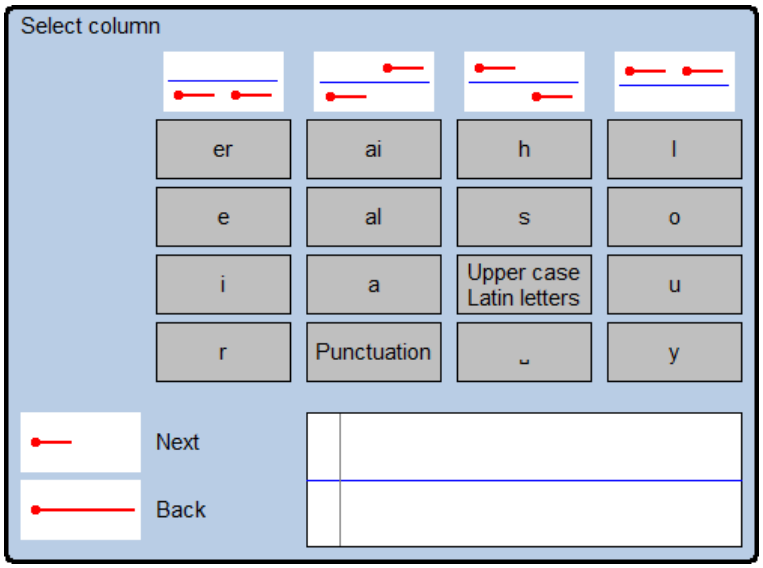


Figure 9.3: Matrix interface.

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 (Figure 9.1, set#1, *Next*) in order to display less probable n-grams. Written text can be erased by producing a long tone (Figure 9.1, set #1, *Back*), in the same manner as in the Direct interface.

List interface

The *List* interface (see Figure 9.4) is controlled by just three simple and easy-to-learn gestures (see Figure 9.1, set #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 (Figure 9.1, set #2, *Key 1 and 2*). A long tone (Figure 9.1, set #2, *Key 3*) 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.

9.3.2 Static Layout

Static layout was designed 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, a static interface was developed which

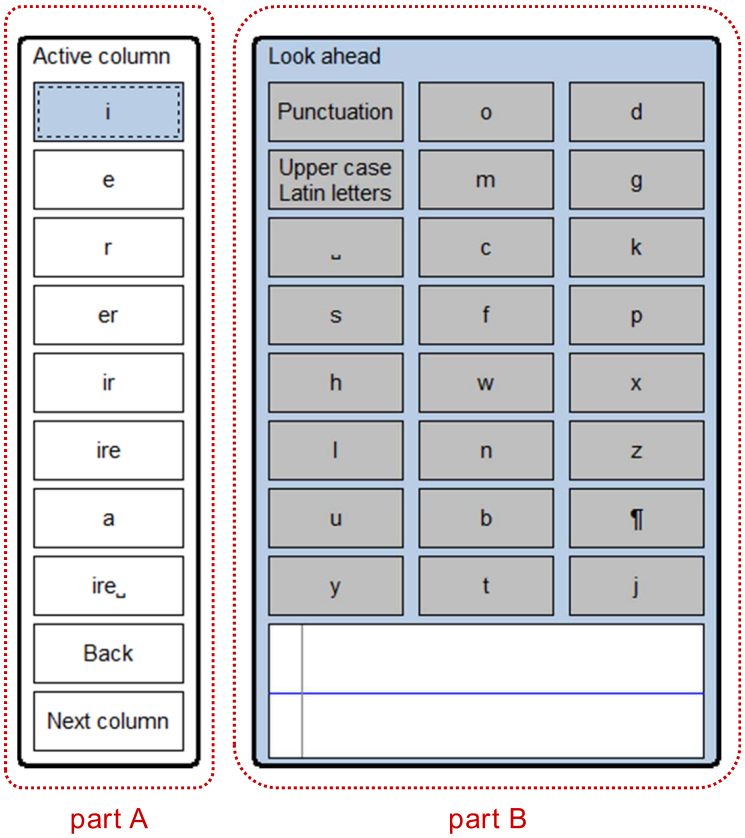


Figure 9.4: List interface. A—active column, B—look ahead matrix.

keeps position of characters and the characters are sorted alphabetically. Time needed to locate a character is then modeled by Hick-Hyman law [71] 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. Moreover, the user may rely on the visual memory.

Binary interface

In the *Binary* interface (see Figure 9.5), the characters are always displayed in an alphabetic order. The Binary interface is based on N-ary search scanning presented in Section 7.3.2, however, the scanning is replaced by direct selection. In the binary interface, characters are selected by splitting the alphabet into two halves and deciding which half is used in the next step. Then the half is split again and again until the desired character is found. Each character is located in following number of steps:

$$steps = \lceil \log_2 N \rceil \tag{9.1}$$

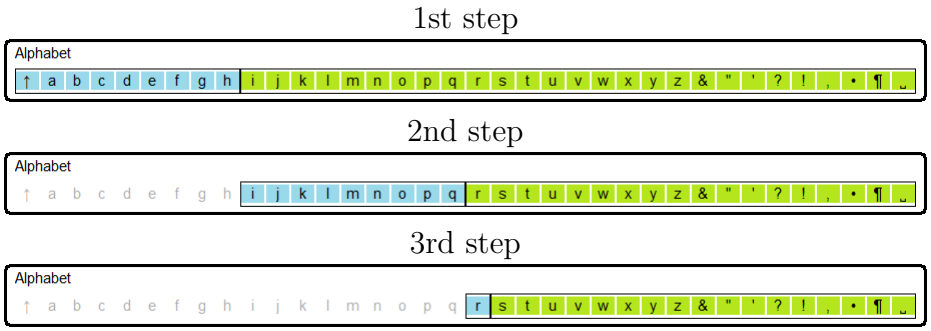


Figure 9.5: Binary interface, typing "*r*" after "*Text ent*".

N is size of the alphabet. In our case, locating a character would require $\lceil \log_2 36 \rceil = 6$ selections as the 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 such 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.

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 9.2, where k is the index of boundary character, p_i is a probability of character i and N is a size of the alphabet.

$$\min_{1<k<N} \left(\left(\frac{1}{2} - \sum_{i=1}^{k-1} p_i \right)^2 + \left(\frac{1}{2} - \sum_{i=k}^{N} p_i \right)^2 \right) \tag{9.2}$$

The Binary interface utilizes only three vocal gestures (see Figure 9.1, set #2) as well as the List interface. Short low tone (*Key 1*) and short high tone (*Key 2*) are used for entering text, while the long tone (*Key 3*) is used for corrections.

An example of user interaction with the Binary interface is depicted in Figure 9.5. 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.

The Binary interface can be easily scaled to a general N-ary interface by adoption N-ary search instead of the binary one. The alphabet would be then split into N groups with balanced probabilities similarly to Equation 9.2. More detailed description and evaluation of the N-ary interfaces is described in Chapter 7.

9.4 Evaluation

In order to evaluate the interfaces, we conducted two user studies. The goal of the first one was to compare all four humming-based interfaces described above (Direct, Matrix, List, and Binary), 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.

9.4.1 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. [112], 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.

Interface	WPM		GPC		Corrections	
	Mean	SD	Mean	SD	Mean	SD
Direct	2.88	.56	1.8	.23	13.0	11.0
Matrix	2.36	.42	1.9	.32	16.1	14.6
List	2.60	.64	3.5	.58	6.4	6.6
Binary	2.34	.36	3.4	.18	14.5	8.5

Table 9.1: Means and standard deviations (SD) of the typing rate (WPM), vocal gesture per character (GPC) rate and total number of corrections.

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 (WPM, GPC rate and number of corrections) are shown in Table 9.1. The number of correction is computed as mean of sum of number of corrections per participant.

The ANOVA test and Scheffé's method [122] were used to find statistically significant differences in mean quantities among interfaces. When comparing mean WPM rates, the Direct interface was significantly faster ($F_{3,64} = 4.20, p < .01$) 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 ($F_{3,64} = 107.7, p < .01$) 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 Figure 9.1, set #1), while the other interfaces only three simple gestures (see Figure 9.1, see #2). As already mentioned in Section 9.3, 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

Interface	Expert 1			Expert 2			Expert 3		
	WPM	GPC	corr	WPM	GPC	corr	WPM	GPC	corr
Direct	5.8	1.5	1	4.8	1.7	8	6.0	1.5	2
Matrix	4.6	1.6	3	4.0	1.9	15	4.6	1.5	2
List	5.0	2.8	0	3.4	3.4	4	5.2	2.9	1
Binary	4.6	3.6	1	3.2	3.6	23	4.0	3.2	10

Table 9.2: Performance of expert users.

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 Figure 9.2 and Figure 9.4). 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.

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 9.2 summarizes WPM, GPC rates and number of corrections for each interface. The speed varied between 3.2 and 6 WPM. Expert 1 and 3 preferred the Direct, while expert 2 preferred Matrix interface.

9.4.2 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 (Figure 9.1, set #2). Binary interface was presented and the participants were asked to enter a phrase.

- **Session 2**: Participants trained more complicated vocal gestures (Figure 9.1, set #1) until required accuracy was achieved. Then the Direct interface was presented to them and they were asked to enter few phrases.

- **Sessions 3–7**: Participants were asked to enter phrases using the Direct interface. On the last day, participants 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.

Participant 1

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.

She spent first and second sessions trying to learn vocal 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 1.6 WPM.

As the participant was unable to produce more complicated gestures, we modified the List interface (see Figure 9.6) 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 3 WPM.

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 (NVVI) is much better than speech for me*"). She reported that after one hour of humming her vocal cords were not tired at all.

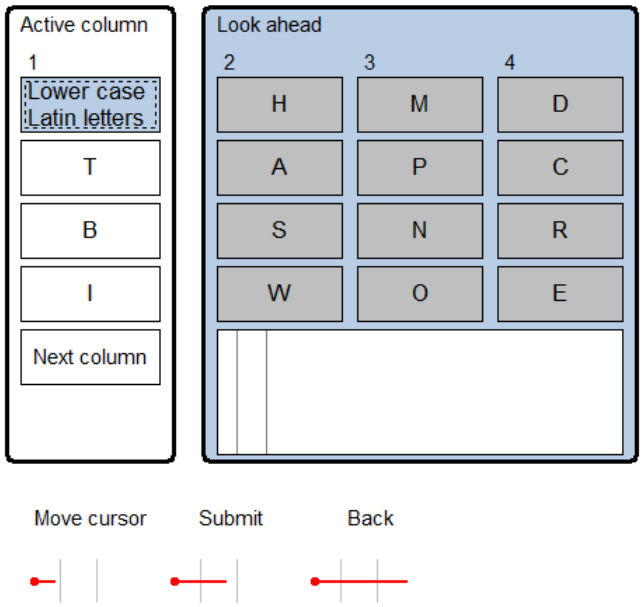


Figure 9.6: Modified List interface.

Participant 2

The participant was a 51 years old man, 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 cords 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 1. As he was not able to produce low and high tone properly, his performance was about 0.2 WPM with a lot of corrections. In the third session, he switched to the modified List interface (see Figure 9.6) as participant 1 and his performance increased rapidly with minimum mistakes. Using this interface and the vocal gestures based on length he reached type rate of 2.8 WPM.

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 cords got tired after 40 minutes of humming.

Participant 3

The participant was a 30 year old man, 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 4.4 WPM. He reported, however, that his current text entry rate achieved by the mouth stick is higher.

Participant 4

Another disabled participant was a 19 years old man, 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 2.4 WPM. After switching to Direct interface, the type rate increased to 4.2 WPM. 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.

9.4.3 Exploring Combination of Humming and Hissing

The case studies described above showed that people with combined motor and speech impairment (participant 1 and 2) had considerable problems when controlling the pitch in the sound of humming. This resulted in development of a redesigned interface which did not require any pitch alterations (List2 interface). However, this interface is slower than the original solutions as less inputs are available. In order to raise the number of inputs, we designed a new set of gestures combining humming and hissing and applied it to the

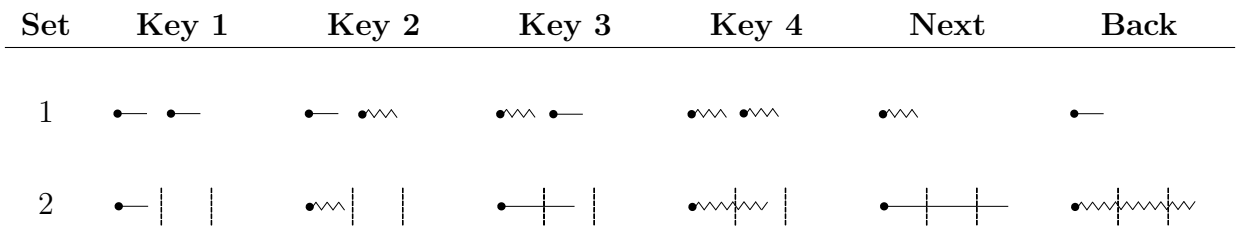


Figure 9.7: Gestures combining humming and hissing. Straight horizontal lines correspond to humming, curly lines to hissing, and dashed vertical lines represent thresholds.

Direct interface. Two gesture sets were designed as shown in Figure 9.7. The straight line represents humming, while the curly line is hissing. Time thresholds are shown as vertical lines.

From the physiological point of view, the hissing and humming sounds originate in different parts of human vocal tract. The humming sound is produced by the vocal folds and the pitch of the sound is determined by their vibration frequency. Hissing is generated in the buccal cavity by using lips and tongue. The vocal chords are not used and hence the pitch of the hissing sound cannot be established. Using both sounds, however, requires a conscious control of the breath by the users.

In order to see how the people with combined motor and speech impairment accept the combination of humming and hissing, we conducted an additional session with participants 1 and 2. The participant 1 tried to type with the gestures from the set #3 for 30 minutes, but her accuracy was too low to be able to type a single phrase. The main problem in this gesture set for her was switching between humming and hissing sounds used in gestures *Keys 1–4*. The participant was not able to change the sound rapidly and accurately enough. Thus, the set #3 was abandoned for this participant. After using the set #4 the accuracy improved, yet not sufficiently. The participant had still problems producing the long hissing sound, which was mapped to the back action. Also, the middle hissing (*Key 4*) sound was not comfortable for her. She was not able to produce hissing sound long enough. Her typing rate did not exceed the previous experiment as this kind of interaction was more physically demanding for her. According to the observations and her comments, she preferred humming only.

The results were different for the participant 2. His speech impairment was not as severe as in the case of the participant 1. He was able to control both sets in a steady rhythm. His speed was comparable to those achieved in the previous sessions, yielding mean type rate at 2.8 WPM with the set #3 and 2.4 WPM with the set #4. He also commented that switching between humming and hissing is challenging and would prefer interfaces employing humming only.

9.4.4 Summary of evaluation

The evaluation focused on assessing four interfaces of Humsher—an adaptive virtual keyboard operated by humming. Three of them (Direct, Matrix and List) used dynamic layout, while the Binary interface used a static layout.

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 several characters could be entered together. The Direct interface was the fastest one with average speed 2.9 WPM achieved by novice and 5.6 WPM by expert users.

Acceptance of the Humsher 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 2.8 and 4.4 WPM. A combination of humming and hissing sound was also tested in order to increase possible range of input signals for people with combined motor and vocal impairments. However, no significant improvement was found when using this combination and a decrease in comfort was observed.

9.5 Measuring Performance

A number of text entry methods use a predictive completion based on letter-level n-gram model. In this section, an optimal length of n-grams stored in such model is investigated. In order to find the length, six different corpora are analyzed, from which a model is built by counting number of vocal gestures needed to enter a text. Based on these numbers, a formula is provided for estimation of words per minute (WPM) rate. The model and the analysis results are verified in an experiment with three experienced users of the keyboard. The model was built for the Direct interface as it was the fastest one relatively well accepted by the users.

9.5.1 Simulation

The main aim of the simulation was to explore how the order of the language model affects efficiency of the Humsher. The other question was how the ideal length of the n-grams differed for various text corpora and languages. We used four publicly available text corpora and three languages:

1. *Dasher.* The corpus is available as a training text for the Dasher[2]. Corpora for various languages is provided—we used English, German and Czech.

[2] `http://www.inference.phy.cam.ac.uk/dasher/Download.html`

2. *AacText.* A crowd-sourced corpus of augmentative and alternative communication (AAC) collected by Vertanen and Kristensson [203].

3. *EnronMobile.* A subset of sentences written by Enron employees on BlackBerry mobile devices published by Vertanen and Kristensson [204].

4. *SmsCorpus.* This public research corpus contains SMS messages collected by National University of Singapore[3].

All text corpora were split to training and test data sets. The training data set was used to train the language model and the test data set was used for text entry simulation. Detailed information about corpora is shown in the Table 9.3.

As already mentioned above, we focused on the Direct interface in which the user can directly choose one of four n-grams from to first column or scan among other columns. Please refer to Section 9.3 for detailed description of the interface.

We used the *gestures per character* (GPC) measure [216] to express the performance of the text entry method. The GPC rate for the Direct interface is defined by the Equation 9.3 where CM is a number of column movements that corresponds to finding the desired column by linear scanning, DS is a number of direct selections that corresponds to selection of desired cell by one of four vocal gestures, and $|T|$ is a length of the test data set. The sum of CM and DS corresponds to total number of vocal gestures in an input stream.

$$GPC = \frac{CM + DS}{|T|} \tag{9.3}$$

The GPC rate is a characteristic measure that is similar to *keystrokes per character* (KSPC) measure and can be used for capturing initial performance of a text entry method [99]. The theoretical text entry speed in terms of *words per minute* (WPM) can be estimated from the CM and DS variables according to the Equation 9.4. The constants a and b represent an average time needed for the column movement and the direct selection respectively. These constants are measured in a subsequent experiment with users described in Section 9.5.2.

$$WPM_{est} = \frac{|T|}{5} \times \frac{60}{aCM + bDS} \tag{9.4}$$

Theoretical GPC for each corpus was analyzed, while changing the order of the language model (i.e. size of n-grams stored in the model) from 1 to 16. As the user actions were simulated by computer, no human errors were taken into account. The resulting dependency of GPC on the order of the language model is depicted in Figure 9.8. The effect is significant for 1^{st}-order to 5^{th}-order model. The difference for higher order models is negligible. Minimal GPC values and corresponding order of the language model are shown

[3] `http://wing.comp.nus.edu.sg:8080/SMSCorpus/history.jsp`; version 2011.12.30

Corpus	Training part size	Words	Unique words	Sentences	Test part size
Dasher (English)	289 KB	51 064	8 676	2 568	30 KB
Dasher (German)	884 KB	122 130	21 951	7 051	53 KB
Dasher (Czech)	419 KB	59 179	21 131	6 148	33 KB
AacText	127 KB	25 125	2 206	3 646	14 KB
EnronMobile	97 KB	18 472	3 041	2 050	10 KB
SmsCorpus	2 748 KB	536 029	27 905	81 836	80 KB

Table 9.3: Detailed information on corpora used in simulation and experiment.

Corpus	Dasher (English)	Dasher (German)	Dasher (Czech)	AacText	EnronMobile	SmsCorpus
Min. GPC	1.23	1.17	1.55	1.11	1.27	1.42
Order	7	11	5	8	6	11

Table 9.4: Minimal theoretical GPC and corresponding order of the language model for all corpora.

in the Table 9.4. Nevertheless, using 6^{th}-order model is sufficient for each corpus as the difference of 6^{th}-order GPC value and the minimal GPC value is always less than 2%.

9.5.2 Experiment

The aim of the experiment was to validate results from the aforementioned analysis and to find out the a and b values for the WPM estimation (see Equation 9.4). In the experiment, 3 able-bodied participants (all men, aged 29–36) took part. They had previous experience with the Humsher as they already participated in an experiment described in Section 9.4.1.

The task in the experiment was to copy two sentences. The independent variables were a corpus and an order of the language model. We used all six corpora and following orders of the language model: 1, 3, 6, and 12. The sentences were unique for each corpus and they were chosen from the test part of each corpora. The experiment was conducted in two trials, in each trial the participant had to copy the two sentences under all conditions (4 orders $\times$ 6 corpora = 24 conditions). Each participant had to copy 96 sentences in the experiment (2 trials $\times$ 2 sentences $\times$ 24 conditions). The experiment took approximately 4 hours per participant. The sequence of corpora the order of the model was randomized for each participant to compensate for learning effects.

Results

The average WPM results for each corpus and order of the language model are shown in Figure 9.9. Similarly, the average GPC results measured in the experiment are shown in

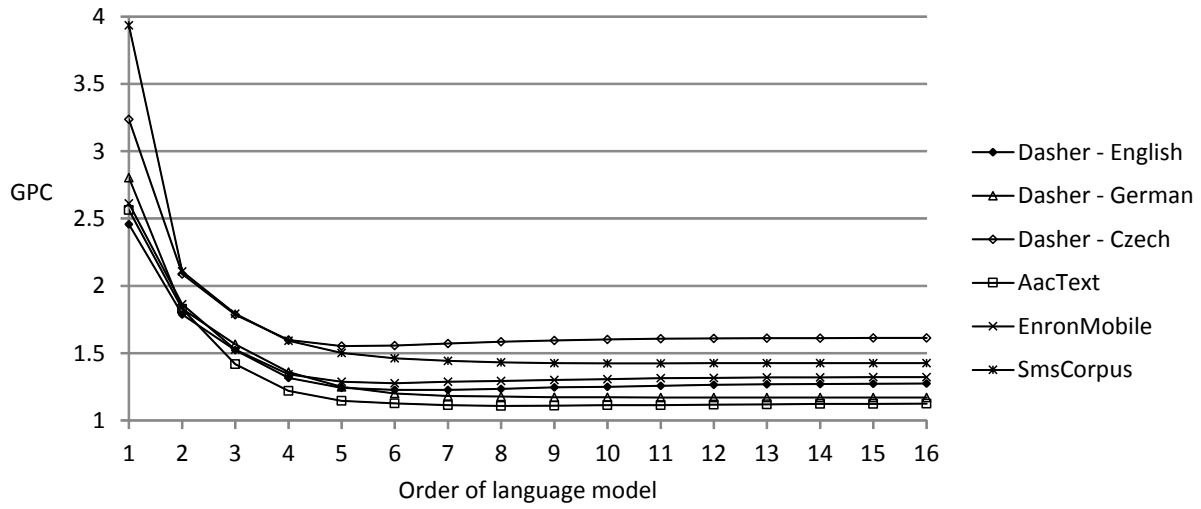


Figure 9.8: Dependence of Humsher text input efficiency in terms of GPC on the order of language model. Simulation results are shown.

Figure 9.10. The best performance was achieved for the *AacText* corpus, probably because it contains limited number of unique words. The peak average entry rate for this corpus was 7.35 WPM at 6^{th}-order of the language model.

Two-factor ANOVA showed a significant interaction between the order and the corpus for GPC ($F_{15,264} = 21.3, p < .001$) and WPM ($F_{15,264} = 6.45, p < .001$) values. Therefore, we evaluated the effect of order on both values separately for each corpus. The Table 9.5 shows significantly different pairs in the GPC and WPM rates denoted by *less than* ($<$) sign. The differences were considered significant on $p < .05$ level.

Two-factor ANOVA for the a and b values showed no interaction between order and corpus, nor significant main effect for the corpus factor. However, significant main effect for the context was found for both values a ($F_{3,264} = 40.5, p < .001$) and b ($F_{3,264} = 15.2, p < .001$). Subsequent ANOVA and post-hoc pairwise comparisons on the order of the model revealed that both values a ($F_{3,254} = 33.1, p < .001$) and b ($F_{3,254} = 14.1, p < .001$) are significantly lower for the 1^{st}-order model. The explanation is simple. Using the 1^{st}-order model ($a = 1.30$s; $b = 1.25$s), the layout of characters is static regardless the already written context. The letter are offered according to their frequency in a corpus. In case of higher-order models ($a = 1.58$s; $b = 1.56$s), on the other hand, a context is used for prediction and it causes different layout of letters (or n-grams) as the user types. Therefore, higher effort is needed to visually locate desired letters, which is reflected in longer time needed to produce a vocal gesture. Note that the difference between the a and b values is minimal. Therefore, we can merge them into one variable c, which simplifies the Equation 9.4 as follows:

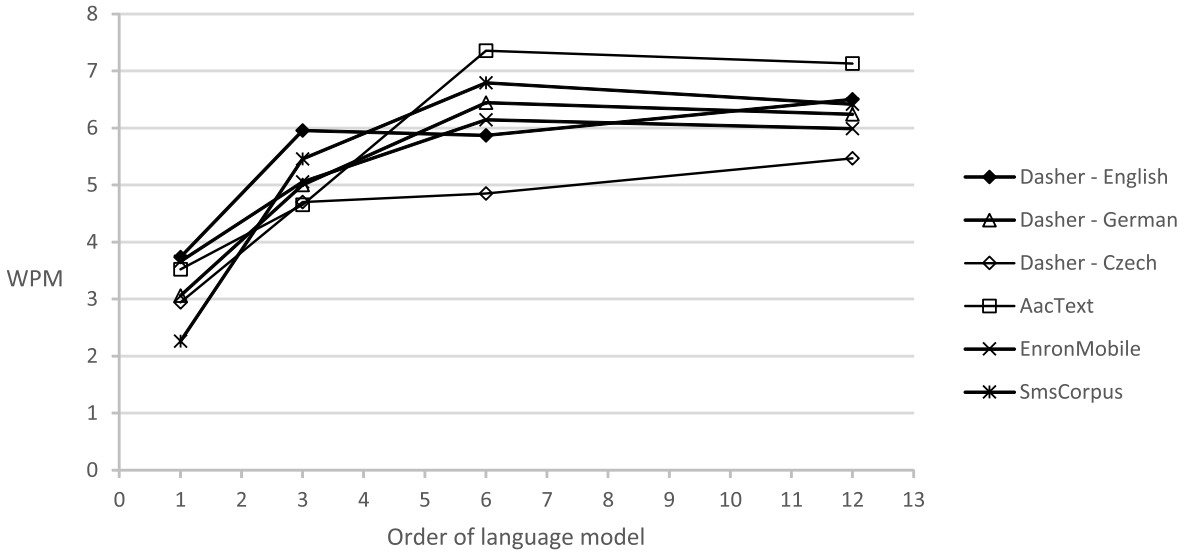


Figure 9.9: Experiment results in terms of WPM entry rate.

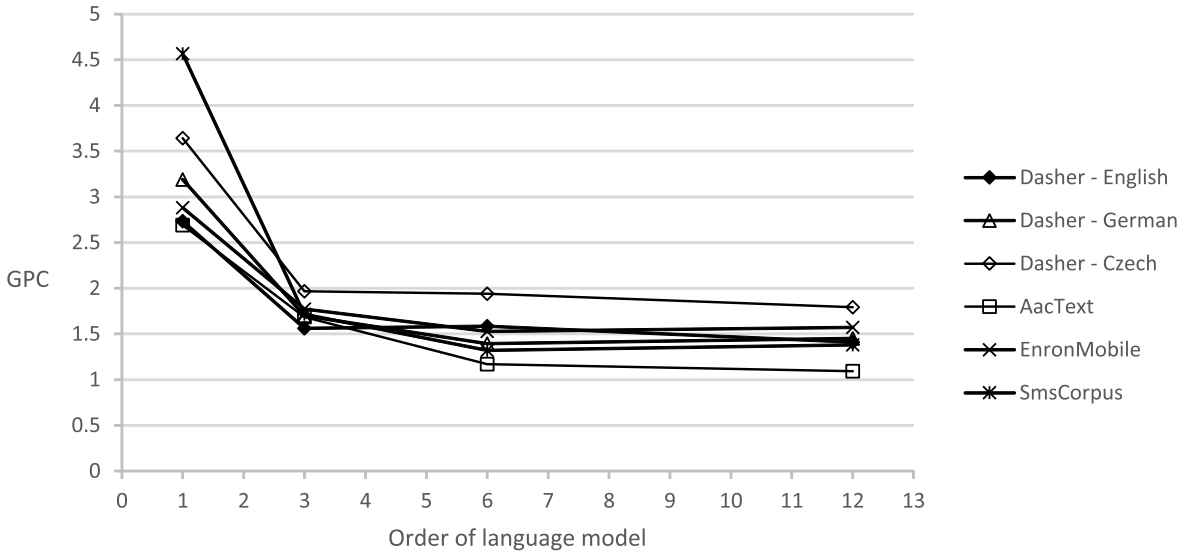


Figure 9.10: Experiment results in terms of GPC rate.

Corpus	GPC	WPM
Dasher (English)	$1 < 3,6,12$	$1 < 3,6,12$
Dasher (German)	$1 < 3,6,12$	$1 < 3 < 6,12$
Dasher (Czech)	$1 < 3,6,12$	$1 < 3,6,12$
AacText	$1 < 3 < 6,12$	$1 < 3 < 6,12$
EnronMobile	$1 < 3,6,12$	$1 < 3 < 6,12$
SmsCorpus	$1 < 3 < 6,12$	$1 < 3 < 6,12$

Table 9.5: Significant differences of the four orders of language model (1,3,6, and 11) in GPC and WPM rates for each corpus.

$$WPM_{est} = \frac{|T|}{5} \times \frac{60}{c(CM + DS)} \tag{9.5}$$

In order to evaluate the model of WPM estimation, a Pearson's product-moment correlation [51] was performed between estimated and measured WPM for all test conditions.

Significant correlations ($p < .001$) were found between the model and the measured values regarding the WPM rate for all three participants: $r_{22} = .91$, $r_{22} = .88$, and $r_{22} = .93$ respectively. As correlations with $r > .90$ are considered very high in experiments with users [101], this result indicates validity of the model for WPM estimation (see Equation 9.4). The correlation coefficient can be even improved by incorporating corrections. The number of correction per character rate was .03, .12, and .10 for each participant respectively. The GPC correlation was even higher ($r_{22} = .98$, $r_{22} = .97$, and $r_{22} = .98$) with high statistical significance ($p < .001$) confirming the correctness of the corpora analysis.

9.5.3 Discussion

Dependence of gesture per character (GPC) rate on order of language model was analyzed for the Direct interface of Humsher. Six publicly available text corpora were used with three languages variants. The analysis results were verified in a controlled experiment with three experienced users. We defined and verified a relation for estimating WPM rate from number of primitive operations.

Producing vocal gestures was faster when no prediction was used and the layout of characters was statically arranged (1^{st}-order language models). However, the prediction used with higher-order models helps to achieve significantly better GPC and WPM rates. We also found that the 6^{th}-order model is adequate for the optimal performance of the keyboard. It is significantly faster than 1^{st}- and 3^{rd}-order model, but is not significantly faster than the 12^{th}-order model. In some cases the 12^{th}-order model even slightly decreases the performance of typing.

Table 9.6: Summary of gesture sets used in the Humsher.

Participant	Speech condition	Applicable sets
1	serious impairment	2a,4*
2	mild impairment	2a,3,4
3	no speech impairment	1,2, (3,4)**
4	no speech impairment	1,2, (3,4)**

* gesture set is applicable only to a certain extent as producing long hissing tones is not accurate enough

** gesture set was not tested with the participant but it was found applicable for people with no speech impairment in pre-studies with able-bodied participants

Although the number of corrections influences the WPM rate, this fact is not incorporated in the model for WPM estimation. Therefore, the WPM estimation formula should be improved by incorporating the error rate or the number of corrections.

9.6 Summary

This chapter aims on predictive keyboard designs operated by humming and hissing. Four novel interfaces were developed and tested with able-bodied and disabled participants. Inclusion of participants with speech impairments showed similar results as in Chapter 8: The NVVI threshold of applicability is lower than that of automatic speech recognition (ASR) which is documented by one of the participants, who have already tried ASR before, but did not work for her.

One of the main contributions of this work is investigation on the applicability of different gesture sets for the participants with the combined impairment. Total of five sets were examined:

- *Set 1* Six humming gestures with tonal alternations
- *Set 2* Three humming gestures with tonal alternations
- *Set 2a* Three humming gestures without tonal alternations based on length only
- *Set 3* Seven gestures combining humming and hissing in single gesture
- *Set 4* Six humming and hissing gestures without combination in a single gesture

Table 9.6 shows the applicability of these gestures sets for motor-impaired people with a range of speech impairments from no impairment to serious impairment. We can see that when properly designed, NVVI can be used with sufficient accuracy by users with a range of speech impairments. This increases the borderline of the target group. For example,

the participant 1 would not be able to use CHANTI as is employs too difficult gestures for her and thus she would be discarded from the study as described in Section 8.4. When the gesture set is carefully designed for her, she is still able to produce required gestures and thus use the Humsher.

While some techniques, such as Dasher [211], offer their users type rates up to 20 WPM, they may not be used by people with severe motor impairments without expensive hardware, such as eye trackers. Humsher, and other text entry methods operated by NVVI do not need specialized hardware and are thus cheap and easy to deploy.

Unlike the ambiguous keyboard CHANTI presented in the previous chapter, Humsher requires no special method for entering out-of-dictionary words. If a word is not present in the language model, it can be still entered relatively rapidly if the word roughly corresponds to statistical distribution of n-grams in the model. The language model used in Humsher is better in modeling languages with complex morphology (e.g. Czech language) than CHANTI. On the other hand, obscure words (e.g., passwords) or words from different languages do not contain probable n-grams and are still quite slow to enter.

Another contribution of this research is identification of optimal order of the language model. In a simulation validated by a controlled experiment, we found 6^{th}-order model as optimal. The main optimization criterion was maximization of prediction accuracy and the secondary criterion was minimization of the size of model. A corpus, from which the model is built, influences the optimal order only marginally.

10 Conclusions

This chapter summarizes results achieved in the thesis and discusses possible directions for the future research.

10.1 Summary of Results

In the thesis, we described how non-verbal vocal input can be applied to text entry methods in order to improve the access to computers for motor-impaired people. As the thesis contributes to the human-computer interaction research field, both NVVI and text entry methods were studied from the human perspective and from the technical point of view. Five aspects listed in Section 1.2 are tackled in the thesis. The most important results from each contribution chapter of the thesis are assigned to appropriate aspect and presented in this summary in order to connect the contributions together and build a bigger and more thorough picture of the studied problem.

10.1.1 Combining text input and NVVI

The NVVI was studied in context of three different keyboards: predictive keyboard, ambiguous keyboard, and scanning keyboard.

The most promising text entry method presented in the thesis is Humsher, a predictive keyboard operated by humming and hissing (Chapter 9). In Humsher, the text is entered using n-grams (groups of letters) with various length sorted according to their probability. The major advantage of the method is the simple and consistent input of out-of-dictionary words. Several designs have been developed and thoroughly iterated with target users.

Use of NVVI and ambiguous keyboard was studied in Chapter 8. It presents CHANTI, a keyboard similar to T9 [88], which uses only four keys. It was evaluated in a longitudinal study with five motor-impaired people. The advantage of the keyboard is the static layout of characters. However, out-of-dictionary words have to be entered by a different method, which considerably decreases text entry rate.

The thesis contributes to scanning keyboards (see Chapter 7) by proposing two novel scanning techniques (N-ary search scanning and row-column scanning on an array). They both ensure static layout of letters even though contextual probability of letters is used. In a subsequent experiment with six scanning keyboards, we found the ternary search scanning keyboard the best among the N-ary keyboards. Surprisingly, some keyboards with dynamic layouts outperformed keyboards with static layouts. The experiment showed that the layout of characters was not found as much important as the scanning technique, which has to be easily predictable by the user.

Prediction and dynamic layouts are often neglected in text entry method research as they always introduce increased cognitive demands on the user. However, these techniques are

helpful when the interaction modality itself is demanding and does not allow the user to react quickly. This often happens in assistive technology and NVVI is not an exception. The results of this thesis support this statement as often dynamic and predictive layouts of characters outperformed static layouts either in terms of entry rate or subjective rating.

While some text entry methods, such as Dasher [211], offer their users type rates up to 20 WPM, they cannot be used by people with severe motor impairments without expensive hardware, such as eye trackers. All text entry methods described in this thesis use standard off-the-shelf hardware and therefore are inexpensive to deploy. They also perform better than the NVVI Keyboard [181] which have the identical hardware requirements and was tested only with able-bodied people.

10.1.2 Text input optimalization

In the thesis, we developed a model for scanning keyboards (see Chapter 7) and for predictive keyboard Humsher (see Chapter 9). Both models can be used for estimation of some performance parameters of new designs and layouts. Both models were validated in an experiment with users. The measured and estimated values yielded high correlation showing the validity of both models. The model of ambiguous keyboard was not described as it has already been developed in the previous work [104].

Moreover, we identified the optimal order of the letter-level statistical language model for Humsher. We found that 6^{th}-order model balances prediction accuracy and the size of the model in an optimal way. The corpus, from which the model is built, influences the optimal order only marginally.

10.1.3 Applicability of NVVI

The applicability of NVVI, was investigated in two studies with disabled people described in Chapter 8 (CHANTI study) and Chapter 9 (Humsher study). We found that participants can be roughly divided into three groups according to their voice abilities.

The first group represents motor-impaired people with severe voice impairment, which hinders them from using NVVI. Vocal input modality (including speech recognition) is not appropriate for them.

The second group represents motor-impaired people with less severe voice impairment who are able to use the NVVI but they cannot use automatic speech recognition (ASR) due to low accuracy of the recognizer. They represent "an optimal target group" of NVVI; i.e. people who are able to speak but who cannot use the ASR. In the thesis, NVVI was found more robust to voice impairments than the ASR. NVVI expands the range of applications of the vocal modality in assistive technologies by an important margin. We may also say that the NVVI threshold of applicability is lower than that of ASR.

The third group represents motor-impaired people without considerable voice impairment. They can use both NVVI and ASR and their performance does not differ significantly from the performance of able-bodied people.

Clearly, NVVI applications should focus on the second group. This group, however, can be broaden and more people can be included by individual design of vocal gestures as shown in the Humsher study (see Chapter 9).

10.1.4 Acceptability of NVVI

Usage of number of pitch-based gestures has been already reported in the literature, however, the subjective attitude towards various aspects of vocal gestures remained unknown. In the thesis, the acceptability of various pitch-based vocal gestures was measured in order to foster the design of NVVI applications (Chapter 6). The results were used in design of pitch-based vocal gestures in Chapters 8 and 9.

The thesis contributes with formulation of four design recommendations for pitch-based gestures. Although the recommendations stemmed from a study with able-bodied people, similar patterns were observed when the NVVI was used by disabled people. Nevertheless, individual approach should be preferred when designing vocal gestures for disabled person owing to heterogeneity of this group.

10.1.5 Accuracy of NVVI

In order to improve the accuracy of the NVVI, a novel method for speech and humming segmentation is proposed in Chapter 5. The method is real-time and is capable of classifying segments of an input audio signal according to their content: speech, humming, or silence. The method is based on computing MFCC and RMS features which are processed by a neural network classifier. The method has been compared to the existing segmentation method based on counting important amplitude changes (IAC method, [182]). The proposed method is speaker-independent, more robust than the IAC method, and the processing latency is lower.

The results from this research were used in NVVI-operated applications described in this thesis (Chapters 6–9) in order to increase the robustness of the interaction.

10.2 Directions of Future Research

The research described in this thesis opens up several possible directions for future research. These directions are outlined in the following paragraphs.

Standardization of a text entry experiment. Comparing different text entry methods is rather complicated as experimental setups vary in each publication. Although ISO 9241-4 describes a standardized experimental setup, no reported evaluation follows this setup as it is tailored for manufacturers of computer keyboards. Thus, we see the need for a standardized experimental setup which would apply to a general text entry method. Defining such setup should be a result of a broad academic discussion as we need to find the tradeoff between resources required and the validity of results. The standardized experimental setup should define namely procedure (number of participants and sessions, length of a session), apparatus (phrases to copy, error correction capability, and error feedback), and dependent variables (type rate and error rate calculations and reporting, subjective evaluation).

Development of text editing functions for motor-impaired people. So far, the research has focused almost exclusively on text entry methods and related studies. Very little work exists on text editors. The methods reported in literature are usually capable only of entering characters and deleting the last typed character. Thus, one of the possibilities of the future work is a research of functions related to the text editing in order to support creation of larger documents. In order to achieve this, the editor should support, for example, cursor movements, insertions, a text selection, formatting commands, or a view moving. Designing such text editor is challenging when the range of possible input signals is low, which is the case of motor-impaired people.

Contextualizing the text input. Text entry methods are usually studied in a simulated context with a predefined language model. In practice, however, the text input is used in a number of situations and the language model should thus adapt to these situations. The language model should be different when, for example, writing an email, or when entering a web address. In case of passwords, for example, the use of language model is problematic as the user typically does not want to update the model with the password to retain the privacy. Another model should be also used when entering numbers, time or date.

Using NVVI for multimodal interaction. NVVI itself has rather limited expressive capabilities as only a few different commands can be produced by the users. Using another modality may increase this number. A future research on combination of NVVI and other modalities (e.g., EEG, EMG, or gaze) can be thus interesting for motor-impaired users.

Applying different interaction modality for text input. Although the text entry methods described in this thesis are designed especially for the NVVI, they can be generalized for use with different and novel input modalities. The methods can be then adapted to the different modality by finding the optimal setting. By adaption, we mean mainly *(i)* finding the optimal number of the input commands, *(ii)* finding the optimal mapping to the

text entry method, and *(iii)* optimizing the layout of the method for the new interaction modality.

Applying the research as a market product. The text entry methods described in the thesis are mostly developed as high-fidelity prototypes. This is sufficient for experimental purposes, however, more research needs to be done to deliver a market-ready release. One of the most important issues in this direction is improvement of immediate usability of the text entry methods developed within the thesis. The immediate usability is often neglected in research of text entry methods although it is quite important factor of user experience. Similar challenge applies to the learning process of NVVI vocal gestures.

10.3 Summary

This thesis has described the use of non-verbal vocal input for the text input by motor-impaired people. The challenges of the thesis, as mentioned in Section 1.2, have been thoroughly investigated. The goal to apply NVVI to novel text entry methods to improve quality of life of motor-impaired has been fulfilled.

Part III

Appendices

Bibliography

[1] Al-Hashimi, S. (2006). Blowtter: A voice-controlled plotter. In *Proceedings of HCI 2006 Engage, The 20th BCS HCI Group conference in co-operation with ACM, vol. 2, London, England*, pages 41–44.

[2] Al Hashimi, S. (2007). Preferences and patterns of paralinguistic voice input to interactive media. In *Proceedings of the 12th international conference on Human-computer interaction: intelligent multimodal interaction environments*, HCI'07, pages 3–12, Berlin, Heidelberg. Springer-Verlag.

[3] Al-Hashimi, S. (2009). Vocal Telekinesis: Towards the Development of Voice-Physical. *Univers. Access Inf. Soc.*, 8:65–75.

[4] Andres, R. O. and Hartung, K. J. (1989). Prediction of head movement time using Fitts' law. *Human Factors*, 31(6):703–713.

[5] Anson, D., George, S., Galup, R., Shea, B., and Vetter, R. (2001). Efficiency of the Chubon versus the QWERTY Keyboard. *Assistive Technology*, 13(1):40–45.

[6] Antona, M. and Stephanidis, C. (2000). An Accessible Word Processor for Disabled People. In *Proceedings of the 7th International Conference on Computers Helping People with Special Needs*, ICCHP 2000, pages 689–696. Österreichische Computer Gesellschaft.

[7] Arnott, J. L. (2006). Text entry in augmentative and alternative communication. In Harbusch, K., Raiha, K.-J., and Tanaka-Ishii, K., editors, *Efficient Text Entry*, number 05382 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.

[8] Ashtiani, B. and MacKenzie, I. S. (2010). Blinkwrite2: an improved text entry method using eye blinks. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, ETRA '10, pages 339–345, New York, NY, USA. ACM.

[9] Baljko, M. and Tam, A. (2006). Indirect text entry using one or two keys. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*, Assets '06, pages 18–25, New York, NY, USA. ACM.

[10] Barrett, M. and San Agustin, J. (2009). Performance evaluation of a low-cost gaze tracker for eye typing. In *Proceedings of the 5th Conference on Communication by Gaze Interaction*, COGAIN '09, pages 13–18.

[11] Beelders, T. R. and Blignaut, P. J. (2012). Measuring the performance of gaze and speech for text input. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, pages 337–340, New York, NY, USA. ACM.

[12] Belatar, M. and Poirier, F. (2008). Text entry for mobile devices and users with severe motor impairments: handiglyph, a primitive shapes based onscreen keyboard. In *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, ASSETS '08, pages 209–216, New York, NY, USA. ACM.

[13] Betke, M., Gips, J., and Fleming, P. (2002). The Camera Mouse: visual tracking of body features to provide computer access for people with severe disabilities. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 10(1):1–10.

[14] Bilmes, J. A., Li, X., Malkin, J., Kilanski, K., Wright, R., Kirchhoff, K., Subramanya, A., Harada, S., Landay, J. A., Dowden, P., and Chizeck, H. (2005). The vocal joystick: a voice-based human-computer interface for individuals with motor impairments. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 995–1002, Morristown, NJ, USA. Association for Computational Linguistics.

[15] Boissiere, P. and Dours, D. (1996). VITIPI: versatile interpretation of text input by persons with impairments. In *Proceedings of the 5th International conference on Computers helping people with special needs. Part I*, ICCHP '96, pages 165–172, Munich, Germany, Germany. R. Oldenbourg Verlag GmbH.

[16] Boissiere, P. and Dours, D. (2003). An overview of existing writing assistance systems. In *Proceedings of the IFRATH Workshop 2003*.

[17] Boissiere, P., Vigouroux, N., Mojahid, M., and Vella, F. (2012). Adaptation of AAC to the Context Communication: A Real Improvement for the User Illustration through the VITIPI Word Completion. In Miesenberger, K., Karshmer, A., Penaz, P., and Zagler, W., editors, *Computers Helping People with Special Needs*, volume 7383 of *Lecture Notes in Computer Science*, pages 451–458. Springer Berlin Heidelberg.

[18] Bolt, R. A. (1980). “put-that-there”: Voice and gesture at the graphics interface. *SIGGRAPH Comput. Graph.*, 14(3):262–270.

[19] Brøndsted, T., Augustensen, S., Fisker, B., Hansen, C., Klitgaard, J., Nielsen, L., and Rasmussen, T. (2001). A system for recognition of hummed tunes. In *Proceedings of the COST G-6 Conference on Digital Audio Effects*, Limerick, Ireland. Department of Computer Science and Information Systems, University of Limerick.

[20] Čadík, M. (2008). Perceptual evaluation of color-to-grayscale image conversions. *Comput. Graph. Forum*, 27(7):1745–1754.

[21] Cattin, G. (1985). *Music of the Middle Ages*. Cambridge University Press.

[22] Chanjaradwichai, S., Punyabukkana, P., and Suchato, A. (2010). Design and evaluation of a non-verbal voice-controlled cursor for point-and-click tasks. In *Proceedings*

of the 4th International Convention on Rehabilitation Engineering & Assistive Technology, iCREATe '10, pages 48:1–48:4, Kaki Bukit TechPark II,, Singapore. Singapore Therapeutic, Assistive & Rehabilitative Technologies (START) Centre.

[23] Chanjaradwichai, S., Punyabukkana, P., and Suchato, A. (2011). Desktop access with non-verbal sound input. In *Proceedings of the 5th International Conference on Rehabilitation Engineering & Assistive Technology*, i-CREATe '11, pages 9:1–9:4. Singapore Therapeutic, Assistive & Rehabilitative Technologies (START) Centre.

[24] Chubon, R. A. and Hester, M. R. (1988). An enhanced standard computer keyboard system for single-finger and typing-stick typing. *Rehabilitation Research and Development*, 25(4):17–24.

[25] Dai, L., Goldman, R., Sears, A., and Lozier, J. (2004). Speech-based cursor control: a study of grid-based solutions. In *Proceedings of the 6th international ACM SIGACCESS conference on Computers and accessibility*, Assets '04, pages 94–101, New York, NY, USA. ACM.

[26] Darragh, J., Witten, I. H., and James, M. (1990). The Reactive Keyboard: a predictive typing aid. *Computer*, 23(11):41–49.

[27] Davis, S. and Mermelstein, P. (1980). Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28:357–366.

[28] Demasco, P. W. and McCoy, K. F. (1992). Generating text from compressed input: an intelligent interface for people with severe motor impairments. *Commun. ACM*, 35(5):68–78.

[29] Dvorak, A., Merrick, N., Dealey, W., and Ford, G. (1936). *Typewriting behavior: psychology applied to teaching and learning typewriting.* American book company.

[30] Evreinova, T., Evreinov, G., and Raisamo, R. (2004). Four-Key Text Entry for Physically Challenged People. In *Adjunct Proceedings of the 8th ERCIM Workshop User Interfaces For All*, UI4ALL 04.

[31] Farwell, L. and Donchin, E. (1988). Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology*, 70(6):510–523.

[32] Fejtová, M., Fejt, J., and Lhotská, L. (2004). Controlling a PC by Eye Movements: The MEMREC Project. In Miesenberger, K., Klaus, J., Zagler, W., and Burger, D., editors, *Computers Helping People with Special Needs*, volume 3118 of *Lecture Notes in Computer Science*, pages 623–623. Springer Berlin / Heidelberg.

[33] Felzer, T. and Freisleben, B. (2002). Hawcos: the "hands-free" wheelchair control system. In *Proceedings of the fifth international ACM conference on Assistive technologies*, Assets '02, pages 127–134, New York, NY, USA. ACM.

[34] Felzer, T., MacKenzie, I., Beckerle, P., and Rinderknecht, S. (2010). Qanti: A Software Tool for Quick Ambiguous Non-standard Text Input. In Miesenberger, K., Klaus, J., Zagler, W., and Karshmer, A., editors, *Computers Helping People with Special Needs*, volume 6180 of *Lecture Notes in Computer Science*, pages 128–135. Springer Berlin Heidelberg.

[35] Felzer, T., MacKenzie, I., and Rinderknecht, S. (2012). DualScribe: A Keyboard Replacement for Those with Friedreichs Ataxia and Related Diseases. In Miesenberger, K., Karshmer, A., Penaz, P., and Zagler, W., editors, *Computers Helping People with Special Needs*, volume 7383 of *Lecture Notes in Computer Science*, pages 431–438. Springer Berlin Heidelberg.

[36] Felzer, T. and Nordmann, R. (2006a). Alternative text entry using different input methods. In *Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*, Assets '06, pages 10–17, New York, NY, USA. ACM.

[37] Felzer, T. and Nordmann, R. (2006b). Speeding up hands-free text entry. In *Proceedings of the 3rd Cambridge Workshop on Universal Access and Assistive Technology*, CWUAAT'06, pages 27–36.

[38] Felzer, T., Nordmann, R., and Rinderknecht, S. (2009). Scanning-based human-computer interaction using intentional muscle contractions. In *Proc. HCI International 2009*, pages 509–518, Berlin, Heidelberg. Springer-Verlag.

[39] Felzer, T. and Rinderknecht, S. (2009). 3dscan: an environment control system supporting persons with severe motor impairments. In *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*, Assets '09, pages 213–214, New York, NY, USA. ACM.

[40] Felzer, T. and Rinderknecht, S. (2011). Using a game controller for text entry to address abilities and disabilities specific to persons with neuromuscular diseases. In *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility*, ASSETS '11, pages 299–300, New York, NY, USA. ACM.

[41] Felzer, T., Strah, B., and Nordmann, R. (2008). Automatic and self-paced scanning for alternative text entry. In *Proceedings of the IASTED International Conference on Telehealth/Assistive Technologies*, Telehealth/AT '08, pages 1–6, Anaheim, CA, USA. ACTA Press.

[42] Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47:381–391.

[43] Fragoulis, D., Avaritsiotis, J., and Papaodysseus, C. (1999). Timbre recognition of single notes using an ARTMAP neural network. In *Proceedings of the 6th IEEE International Conference on Electronics, Circuits and Systems*, ICECS '99, pages 1009–1012, Washington, DC, USA. IEEE Computer Society.

[44] Fu, Y. and Huang, T. (2007). hmouse: Head tracking driven virtual computer mouse. In *Applications of Computer Vision, 2007. WACV '07. IEEE Workshop on*, pages 30–30.

[45] Garay-Vitoria, N. and Abascal, J. (2006). Text prediction systems: a survey. *Univers. Access Inf. Soc.*, 4(3):188–203.

[46] Ghias, A., Logan, J., Chamberlin, D., and Smith, B. C. (1995). Query by humming: musical information retrieval in an audio database. In *Proceedings of the third ACM international conference on Multimedia*, MULTIMEDIA '95, pages 231–236, New York, NY, USA. ACM.

[47] Gong, J. and Tarasewich, P. (2005). Alphabetically constrained keypad designs for text entry on mobile devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 211–220, New York, NY, USA. ACM.

[48] Google (2013). Ngram viewer.

[49] Gorodnichy, D., Malik, S., and Roth, G. (2002). Nouse 'use your nose as a mouse' – a new technology for hands-free games and interfaces. In *Proceedings of the International Conference on Vision Interface*, VI '02, pages 254–361. National Research Council Canada.

[50] Grabe, M. and Grabe, C. (2004). *Integrating Technology For Meaningful Learning.* Houghton Mifflin Company.

[51] Graziano, A. M. and Raulin, M. L. (2012). *Research Methods: A Process of Inquiry.* Pearson, 8th edition.

[52] Grover, D. L., King, M. T., and Kushler, C. A. (1998). Reduced keyboard disambiguating computer. US Patent No 5818437.

[53] Gutowitz, H. (2003). Barriers to adoption of dictionary-based text-entry methods: a field study. In *Proceedings of the 2003 EACL Workshop on Language Modeling for Text Entry Methods*, TextEntry '03, pages 33–41, Stroudsburg, PA, USA. Association for Computational Linguistics.

[54] Hacihabiboglu, H. and Canagarajah, C. (2002). Musical instrument recognition with wavelet envelopes. In *Proceedings of Forum Acusticum (CD-ROM).*

[55] Hämäläinen, P., Mäki-Patola, T., Pulkki, V., and Airas, M. (2004). Musical computer games played by singing. In Evangelista, G. and Testa, I., editors, *Proceedings of 7th International Conference on Digital Audio Effects, Naples, Italy*, pages 367–371.

[56] Hamidi, F., Baljko, M., Livingston, N., and Spalteholz, L. (2010). Canspeak: A customizable speech interface for people with dysarthric speech. In Miesenberger, K., Klaus, J., Zagler, W., and Karshmer, A., editors, *Computers Helping People with Special Needs*, volume 6179 of *Lecture Notes in Computer Science*, pages 605–612. Springer Berlin Heidelberg.

[57] Hansen, J. P., Johansen, A. S., Hansen, D. W., Itoh, K., and Mashino, S. (2003). Language Technology in a Predictive, Restricted On-screen Keyboard with Ambiguous Layout for Severely Disabled People. In *Proceedings of EACL 2003 Workshop on Language Modeling for Text Entry Methods*, EACL '03.

[58] Hansen, J. P., Tørning, K., Johansen, A. S., Itoh, K., and Aoki, H. (2004). Gaze typing compared with input by head and hand. In *Proceedings of the 2004 symposium on Eye tracking research & applications*, ETRA '04, pages 131–138, New York, NY, USA. ACM.

[59] Harada, S., Landay, J. A., Malkin, J., Li, X., and Bilmes, J. A. (2006). The Vocal Joystick: Evaluation of Voice-Based Cursor Control Techniques. In *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '06, pages 197–204, New York, NY, USA. ACM.

[60] Harada, S., Saponas, T. S., and Landay, J. A. (2007a). Voicepen: augmenting pen input with simultaneous non-linguisitic vocalization. In *Proceedings of the 9th international conference on Multimodal interfaces*, ICMI '07, pages 178–185, New York, NY, USA. ACM.

[61] Harada, S., Wobbrock, J. O., and Landay, J. A. (2007b). Voicedraw: a hands-free voice-driven drawing application for people with motor impairments. In *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, ASSETS '07, pages 27–34, New York, NY, USA. ACM.

[62] Harada, S., Wobbrock, J. O., and Landay, J. A. (2011). Voice games: investigation into the use of non-speech voice input for making computer games more accessible. In *Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part I*, INTERACT'11, pages 11–29, Berlin, Heidelberg. Springer-Verlag.

[63] Harada, S., Wobbrock, J. O., Malkin, J., Bilmes, J. A., and Landay, J. A. (2009). Longitudinal study of people learning to use continuous voice-based cursor control. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 347–356, New York, NY, USA. ACM.

[64] Harbusch, K. and Kühn, M. (2003). An evaluation study of two-button scanning with ambiguous keyboards. In *Proceedings of the 7th European Conference for the Advancement of Assistive Technology*, AAATE'03, pages 954–958.

[65] Harbusch, K. and Kühn, M. (2003). Towards an adaptive communication aid with text input from ambiguous keyboards. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics - Volume 2*, EACL '03, pages 207–210, Stroudsburg, PA, USA. Association for Computational Linguistics.

[66] Hild, K. E., Orhan, U., Erdogmus, D., Roark, B., Oken, B., Purwar, S., Nezamfar, H., and Fried-Oken, M. (2011). An erp-based brain-computer interface for text entry using rapid serial visual presentation and language modeling. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Systems Demonstrations*, HLT '11, pages 38–43, Stroudsburg, PA, USA. Association for Computational Linguistics.

[67] Holzinger, A., Searle, G., Kleinberger, T., Seffah, A., and Javahery, H. (2008). Investigating usability metrics for the design and development of applications for the elderly. In *Proceedings of the 11th international conference on Computers Helping People with Special Needs*, ICCHP '08, pages 98–105, Berlin, Heidelberg. Springer-Verlag.

[68] Holzinger, A., Thimbleby, H., and Beale, R. (2010). Editorial: Human-computer interaction for medicine and health care (hci4med): Towards making information usable. *Int. J. Hum.-Comput. Stud.*, 68(6):325–327.

[69] House, B., Malkin, J., and Bilmes, J. (2009). The voicebot: a voice controlled robot arm. In *Proceedings of the 27th international conference on Human factors in computing systems*, CHI '09, pages 183–192, New York, NY, USA. ACM.

[70] Huffman, D. (1952). A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101.

[71] Hyman, R. (1953). Stimulus information as a determinant of reaction time. *Experimental Psychology*, 45(3):188–196.

[72] Igarashi, T. and Hughes, J. F. (2001). Voice as sound: using non-verbal voice input for interactive control. In *UIST '01: Proc 14th Annual ACM Symp on User Interface Software and Technology*, pages 155–156, New York, NY, USA. ACM Press.

[73] Isokoski, P. (2000). Text input methods for eye trackers using off-screen targets. In *Proceedings of the 2000 symposium on Eye tracking research & applications*, ETRA '00, pages 15–21, New York, NY, USA. ACM.

[74] Isokoski, P. and Raisamo, R. (2000). Device independent text input: a rationale and an example. In *Proceedings of the working conference on Advanced visual interfaces*, AVI '00, pages 76–83, New York, NY, USA. ACM.

[75] Jacob, R. J. K. (1991). The use of eye movements in human-computer interaction techniques: what you look at is what you get. *ACM Trans. Inf. Syst.*, 9(2):152–169.

[76] Johansen, A. and Hansen, J. (2003). Augmentative and alternative communication: The future of text on the move. In Carbonell, N. and Stephanidis, C., editors, *Universal Access Theoretical Perspectives, Practice, and Experience*, volume 2615 of *Lecture Notes in Computer Science*, pages 425–441. Springer Berlin Heidelberg.

[77] Jones, P. E. (1998). Virtual keyboard with scanning and augmented by prediction. In *Proceedings of the 2nd European conference on disability, virtual reality and associated technologies*, ECDVRAT '98, pages 45–51. The University of Reading.

[78] Juang, B. and Rabiner, L. R. (2005). Automatic speech recognition–a brief history of the technology development. Technical report, Georgia Institute of Technology.

[79] Judge, S. and Friday, M. (2011). Ambiguous keyboards for AAC. *Journal of Assistive Technologies*, 5(4):249–256.

[80] Kim, B.-W., Choi, D.-L., and Lee, Y.-J. (2007). Speech/music discrimination using mel-cepstrum modulation energy. In Matouek, V. and Mautner, P., editors, *Text, Speech and Dialogue*, volume 4629 of *Lecture Notes in Computer Science*, pages 406–414. Springer Berlin / Heidelberg.

[81] Kim, H. and Kim, Y.-H. (2009). Optimal designs of ambiguous mobile keypad with alphabetical constraints. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, GECCO '09, pages 1931–1932, New York, NY, USA. ACM.

[82] Kjeldsen, R. (2007). An On-Screen Keyboard for Users with Poor Pointer Control. In Stephanidis, C., editor, *Universal Access in Human-Computer Interaction. Applications and Services*, volume 4556 of *Lecture Notes in Computer Science*, pages 339–348. Springer Berlin Heidelberg.

[83] Koester, H. H. and Levine, S. P. (1994). Learning and performance of able-bodied individuals using scanning systems with and without word prediction. *Assistive Technology*, 6(1):42–53. PMID: 10147209.

[84] Költringer, T., Isokoski, P., and Grechenig, T. (2007). Twostick: writing with a game controller. In *Proceedings of Graphics Interface 2007*, GI '07, pages 103–110, New York, NY, USA. ACM.

[85] Kristensson, P. O. and Vertanen, K. (2012). The potential of dwell-free eye-typing for fast assistive gaze communication. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, pages 241–244, New York, NY, USA. ACM.

[86] Kühn, M. and Garbe, J. (2001). Predictive and highly ambiguous typing for a severely speech and motion impaired user. In *Universal Access in Human-Computer Interaction.*

[87] Kurniawan, S. (2008). Older people and mobile phones: A multi-method investigation. *International Journal of Human-Computer Studies*, 66(12):889–901.

[88] Kushler, C. (1998). *AAC: Using a Reduced Keyboard.*

[89] Lafi, S. M. and Hock, S. K. B. (2012). An Adaptive Text Entry Method Based On Single-Key Minimal Scan Matrix for People with Severe Motor Disabilities. *Scientific and Engineering Research*, 3(8):1–5.

[90] Ledda, P., Chalmers, A., Troscianko, T., and Seetzen, H. (2005). Evaluation of tone mapping operators using a high dynamic range display. *ACM Trans. Graph.*, 24:640–648.

[91] Lesher, G., Moulton, B., and Higginbotham, D. (1998a). Optimal character arrangements for ambiguous keyboards. *Rehabilitation Engineering, IEEE Transactions on*, 6(4):415–423.

[92] Lesher, G., Moulton, B., and Higginbotham, D. J. (1998b). Techniques for augmenting scanning communication. *Augmentative and Alternative Communication*, 14(2):81–101.

[93] Likert, R. (1932). A technique for the measurement of attitudes. *Archives of Psychology*, 22(140):5–55.

[94] Lin, Y.-L., Chen, M.-C., Wu, Y.-P., Yeh, Y.-M., and Wang, H.-P. (2007). A Flexible On-Screen Keyboard: Dynamically Adapting for Individuals Needs. In Stephanidis, C., editor, *Universal Access in Human-Computer Interaction. Applications and Services*, volume 4556 of *Lecture Notes in Computer Science*, pages 371–379. Springer Berlin Heidelberg.

[95] Lin, Y.-L., Wu, T.-F., Chen, M.-C., Yeh, Y.-M., and Wang, H.-P. (2008). Designing a scanning on-screen keyboard for people with severe motor disabilities. In Miesenberger, K., Klaus, J., Zagler, W., and Karshmer, A., editors, *Computers Helping People with Special Needs*, volume 5105 of *Lecture Notes in Computer Science*, pages 1184–1187. Springer Berlin Heidelberg.

[96] Loewenich, F. and Maire, F. (2007). Hands-free mouse-pointer manipulation using motion-tracking and speech recognition. In *Proceedings of the 19th Australasian conference on Computer-Human Interaction: Entertaining User Interfaces*, OZCHI '07, pages 295–302, New York, NY, USA. ACM.

[97] Lu, G. and Hankinson, T. (1998). A technique towards automatic audio classification and retrieval. In *Proceedings of the 4th International Conference on Signal Processing*, volume 2 of *ICSP '98*, pages 1142–1145. IEEE Computer Society.

[98] Lyons, K., Starner, T., Plaisted, D., Fusia, J., Lyons, A., Drew, A., and Looney, E. W. (2004). Twiddler typing: one-handed chording text entry for mobile phones. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 671–678, New York, NY, USA. ACM.

[99] MacKenzie, I. (2002). KSPC (Keystrokes per Character) as a Characteristic of Text Entry Techniques. In Patern, F., editor, *Human Computer Interaction with Mobile Devices*, volume 2411 of *Lecture Notes in Computer Science*, pages 195–210. Springer Berlin Heidelberg.

[100] MacKenzie, I. (2012). Modeling Text Input for Single-Switch Scanning. In Miesenberger, K., Karshmer, A., Penaz, P., and Zagler, W., editors, *Computers Helping People with Special Needs*, volume 7383 of *Lecture Notes in Computer Science*, pages 423–430. Springer Berlin Heidelberg.

[101] MacKenzie, I. S. (1992). Movement time prediction in human-computer interfaces. In *Proceedings of Graphics Interface*, volume 92, pages 140–150.

[102] MacKenzie, I. S. (2007). Evaluation of text entry techniques. In MacKenzie, I. S. and K.Tanaka-Ishii, editors, *Text Entry Systems: Mobility, Accessibility, Universality*, pages 75–101. Morgan Kaufmann.

[103] MacKenzie, I. S. (2009). The one-key challenge: searching for a fast one-key text entry method. In *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*, Assets '09, pages 91–98, New York, NY, USA. ACM.

[104] MacKenzie, I. S. and Felzer, T. (2010). Sak: Scanning ambiguous keyboard for efficient one-key text entry. *ACM Trans. Comput.-Hum. Interact.*, 17(3):11:1–11:39.

[105] MacKenzie, I. S., Kober, H., Smith, D., Jones, T., and Skepner, E. (2001). LetterWise: prefix-based disambiguation for mobile text input. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, pages 111–120, New York, NY, USA. ACM.

[106] MacKenzie, I. S. and Soukoreff, R. W. (2003). Phrase sets for evaluating text entry techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, pages 754–755, New York, NY, USA. ACM.

[107] MacKenzie, I. S., Soukoreff, R. W., and Helga, J. (2011). 1 thumb, 4 buttons, 20 words per minute: design and evaluation of h4-writer. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 471–480, New York, NY, USA. ACM.

[108] MacKenzie, I. S. and Teather, R. J. (2012). Fittstilt: the application of fitts' law to tilt-based interaction. In *Proceedings of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design*, NordiCHI '12, pages 568–577, New York, NY, USA. ACM.

[109] MacKenzie, I. S. and Zhang, S. (1997). The immediate usability of Graffiti. In *Proceedings of Graphics Interface*, pages 129–137. Canadian Information Processing Society.

[110] MacKenzie, I. S. and Zhang, S. X. (1999). The design and evaluation of a high-performance soft keyboard. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '99, pages 25–31, New York, NY, USA. ACM.

[111] MacKenzie, I. S. and Zhang, X. (2008). Eye typing using word and letter prediction and a fixation algorithm. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, ETRA '08, pages 55–58, New York, NY, USA. ACM.

[112] Mahmud, M., Sporka, A. J., Kurniawan, S. H., and Slavík, P. (2007). A comparative longitudinal study of non-verbal mouse pointer. In *Proceedings of INTERACT 2007, Rio de Janeiro, Brazil; Lecture Notes in Computer Science (LNCS) vol. 4663, Part II.*, pages 489–502. Springer-Verlag Berlin Heidelberg.

[113] Majaranta, P., Ahola, U.-K., and Špakov, O. (2009). Fast gaze typing with an adjustable dwell time. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 357–360, New York, NY, USA. ACM.

[114] Majaranta, P., MacKenzie, I., Aula, A., and Räihä, K.-J. (2006). Effects of feedback and dwell time on eye typing speed and accuracy. *Universal Access in the Information Society*, 5(2):199–208.

[115] Majaranta, P. and Räihä, K.-J. (2002). Twenty years of eye typing: systems and design issues. In *Proceedings of the 2002 symposium on Eye tracking research & applications*, ETRA '02, pages 15–22, New York, NY, USA. ACM.

[116] Malkin, J., House, B., and Bilmes, J. (2007). Control of simulated arm with the Vocal Joystick. In *CHI 2007 Workshop on Striking a C[h]ord: Vocal Interaction in Assistive Technologies, Games, and More.*

[117] Manaris, B. and Harkreader, A. (1998). SUITEKeys: a speech understanding interface for the motor-control challenged. In *Proceedings of the third international ACM conference on Assistive technologies*, Assets '98, pages 108–115, New York, NY, USA. ACM.

[118] Manaris, B. and McCauley, R. (2001). An Intelligent Interface for Keyboard and Mouse Control – Providing Full Access to PC Functionality via Speech. In *Proceedings of 14th International Florida AI Research Symposium*, FLAIRS-01, pages 182–188. AAAI Press.

[119] Martin, A., Charlet, D., and Mauuary, L. (2001). Robust speech/non-speech detection using lda applied to mfcc. In *Proceedings of the 2001 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1 of *ICASSP '01*, pages 237–240.

[120] Martin, B., Isokoski, P., Karmann, G., and Rollinger, T. (2012). Continuous edgewrite: dictionary-based disambiguation instead of explicit segmentation by the user. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pages 357–364, New York, NY, USA. ACM.

[121] Matias, E., MacKenzie, I. S., and Buxton, W. (1993). Half-QWERTY: a one-handed keyboard facilitating skill transfer from QWERTY. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pages 88–94, New York, NY, USA. ACM.

[122] Maxwell, S. and Delaney, H. (2004). *Designing Experiments and Analyzing Data: A Model Comparison Perspective*. Lawrence Erlbaum Associates.

[123] Merlin, B. and Raynal, M. (2010). Evaluation of SpreadKey System with Motor Impaired Users. In Miesenberger, K., Klaus, J., Zagler, W., and Karshmer, A., editors, *Computers Helping People with Special Needs*, volume 6180 of *Lecture Notes in Computer Science*, pages 112–119. Springer Berlin Heidelberg.

[124] Mihara, Y., Shibayama, E., and Takahashi, S. (2005). The migratory cursor: accurate speech-based cursor movement by moving multiple ghost cursors using non-verbal vocalizations. In *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*, Assets '05, pages 76–83, New York, NY, USA. ACM.

[125] Miotto, A., Lessiter, J., and Freeman, J. (2009). Vital Mind: an Interactive set-top box platform for cognitive training applications. In *Virtual Rehabilitation International Conference, 2009*, pages 207–207. IEEE Computer Society.

[126] Miró, J. and Bernabeu, P. (2008). Text entry system based on a minimal scan matrix for severely physically handicapped people. In Miesenberger, K., Klaus, J., Zagler, W., and Karshmer, A., editors, *Computers Helping People with Special Needs*, volume 5105 of *Lecture Notes in Computer Science*, pages 1216–1219. Springer Berlin Heidelberg.

[127] Miró-Borrás, J. and Bernabeu-Soler, P. (2008). E-everything for all: Text entry for people with severe motor disabilities. In *Proceedings of the Collaborative Electronic Communications and eCommerce Technology and Research Iberoamerica*, 6th CollECTeR, pages 1–7.

[128] Miró-Borrás, J. and Bernabeu-Soler, P. (2009). Text entry in the e-commerce age: two proposals for the severely handicapped. *J. Theor. Appl. Electron. Commer. Res.*, 4(1):101–112.

[129] Miró-Borrás, J., Bernabeu-Soler, P., Llinares, R., and Igual, J. (2010a). Evaluation of an ambiguous-keyboard prototype scanning-system with word and character disambiguation. In *Proceedings of the 24th BCS Interaction Specialist Group Conference*, BCS '10, pages 403–411, Swinton, UK, UK. British Computer Society.

[130] Miró-Borrás, J., Bernabeu-Soler, P., Llinares, R., and Igual, J. (2010b). A prototype scanning system with an ambiguous keyboard and a predictive disambiguation algorithm. In Miesenberger, K., Klaus, J., Zagler, W., and Karshmer, A., editors, *Computers Helping People with Special Needs*, volume 6180 of *Lecture Notes in Computer Science*, pages 136–139. Springer Berlin Heidelberg.

[131] Molina, A. J., Rivera, O., Gómez, I., Merino, M., and Ropero, J. (2011). Comparison Among Ambiguous Virtual Keyboards For People With Severe Motor Disabilities. *Modern Engineering Research*, 1(2):288–305.

[132] Morimoto, C. H. and Amir, A. (2010). Context switching for fast key selection in text entry applications. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, ETRA '10, pages 271–274, New York, NY, USA. ACM.

[133] Mourouzis, A., Boutsakis, E., Ntoa, S., Antona, M., and Stephanidis, C. (2007). An accessible and usable soft keyboard. In Stephanidis, C., editor, *Universal Access in Human-Computer Interaction. Ambient Interaction*, volume 4555 of *Lecture Notes in Computer Science*, pages 961–970. Springer Berlin Heidelberg.

[134] Newell, A. and Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the law of practice. In Anderson, J. R., editor, *Cognitive skills and their acquisition*, pages 1–55. Lawerence Erlbaum Associates.

[135] Nigay, L. (2004). Design space for multimodal interaction. In *Building the Information Society*, volume 156 of *IFIP International Federation for Information Processing*, pages 403–408. Springer Boston.

[136] Norman, D. A. and Fisher, D. (1982). Why alphabetic keyboards are not easy to use: Keyboard layout doesn't much matter. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 24(5):509–519.

[137] Norte, S. and Lobo, F. G. (2007). A virtual logo keyboard for people with motor disabilities. In *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, ITiCSE '07, pages 111–115, New York, NY, USA. ACM.

[138] Nouza, J., Nouza, T., and erva, P. (2005). A Multi-Functional Voice-Control Aid for Disabled Persons. In *Proceedings of the 10th International Conference on Speech and Computer*, SPECOM '05, pages 715–718, Patras, Greece.

[139] Nouza, J., Zdansky, J., Cerva, P., and Silovsky, J. (2010). Challenges in Speech Processing of Slavic Languages (Case Studies in Speech Recognition of Czech and Slovak). In Esposito, A., Campbell, N., Vogel, C., Hussain, A., and Nijholt, A., editors, *Development of Multimodal Interfaces: Active Listening and Synchrony*, volume 5967 of *Lecture Notes in Computer Science*, pages 225–241. Springer Berlin Heidelberg.

[140] Ntoa, S., Margetis, G., Antona, M., and Stephanidis, C. (2013). Scanning-based interaction techniques for motor impaired users. In Kouroupetroglou, G., editor, *Assistive Technologies and Computer Access for Motor Disabilities*, chapter 3. IGI Global.

[141] office, P. (1855). *Reference index of patents of invention, from 1617 to 1852, by . Woodcroft.* Oxford University.

[142] Orhan, U., Hild, K., Erdogmus, D., Roark, B., Oken, B., and Fried-Oken, M. (2012). RSVP keyboard: An EEG based typing interface. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 645–648.

[143] Oviatt, S. (2003). *The human-computer interaction handbook*, chapter Multimodal interfaces, pages 286–304. L. Erlbaum Associates Inc., Hillsdale, NJ, USA.

[144] Oviatt, S., Cohen, P., Wu, L., Duncan, L., Suhm, B., Bers, J., Holzman, T., Winograd, T., Landay, J., Larson, J., and Ferro, D. (2000). Designing the user interface for multimodal speech and pen-based gesture applications: State-of-the-art systems and future research directions. *Human-Computer Interaction*, 15(4):263–322.

[145] Panwar, P., Sarcar, S., and Samanta, D. (2012). Eyeboard: A fast and accurate eye gaze-based text entry system. In *Intelligent Human Computer Interaction (IHCI), 2012 4th International Conference on*, pages 1–8. IEEE.

[146] Papoulis, A. (1984). *Probability, Random Variables, and Stochastic Processes*, chapter Bernoulli Trials, pages 57–63. McGraw-Hill Inc., New York, 2nd edition.

[147] Pavlovych, A. and Stuerzlinger, W. (2003). Less-Tap: A fast and easy-to-learn text input technique for phones. In *Graphics Interface*, pages 97–104.

[148] Peng, H., Long, F., and Ding, C. (2005). Feature selection based on mutual information: Criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:1226–1238.

[149] Perera, D., Eales, R. T. J., and Blashki, K. (2007). Voice art: investigating paralinguistic voice as a mode of interaction to create visual art. In *Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...but not as we know it - Volume 2*, BCS-HCI '07, pages 87–90, Swinton, UK, UK. British Computer Society.

[150] Perera, D., Jim Eales, R., and Blashki, K. (2009). Supporting the creative drive: investigating paralinguistic voice as a mode of interaction for artists with upper limb disabilities. *Universal Access in the Information Society*, 8(2):77–88.

[151] Perlin, K. (1998). Quikwriting: continuous stylus-based text entry. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, UIST '98, pages 215–216, New York, NY, USA. ACM.

[152] Prabhu, V. and Prasad, G. (2011). Designing a virtual keyboard with multi-modal access for people with disabilities. In *Information and Communication Technologies (WICT), 2011 World Congress on*, pages 1133–1138.

[153] Prechelt, L. and Typke, R. (2001). An interface for melody input. *ACM Transactions on Computer-Human Interaction*, 8(2):133–149.

[154] Pruthi, T., Chhatpar, S., Ngia, L., Brown, J., and Harris, J. (2008). *Method for non-speech vocalization to control UGVs using humming.* AUVSIs Unmanned Systems North America 2008. www.think-a-move.com/pdfs/AUVSIJune2008.pdf.

[155] Rabiner, L. (1977). On the use of autocorrelation analysis for pitch detection. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 25(1):24–33.

[156] Räihä, K.-J. and Ovaska, S. (2012). An exploratory study of eye typing fundamentals: dwell time, text entry rate, errors, and workload. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, pages 3001–3010, New York, NY, USA. ACM.

[157] Rivera, O., Molina, A., Gómez, I. M., and Merino, M. (2009). A Study of Two-inputs Scanning Methods to Enhance the Communication Rate. In Emiliani, P., Burzagli, L., Como, A., Gabbanini, F., and Salminen, A.-L., editors, *Assistive Technology from Adapted Equipment to Inclusive Environments*, volume 25 of *AAATE 2009*, pages 132–137. IOS Press.

[158] Roark, B., Beckley, R., Gibbons, C., and Fried-Oken, M. (2013). Huffman scanning: Using language models within fixed-grid keyboard emulation. *Computer Speech & Language*, 27(6):1212–1234.

[159] Roark, B., de Villiers, J., Gibbons, C., and Fried-Oken, M. (2010). Scanning methods and language modeling for binary switch typing. In *Proceedings of the NAACL HLT 2010 Workshop on Speech and Language Processing for Assistive Technologies*, SLPAT '10, pages 28–36, Stroudsburg, PA, USA. Association for Computational Linguistics.

[160] Ronzhin, A. and Karpov, A. (2005). Assistive multimodal system based on speech recognition and head tracking. In *Proceedings of 13th European Signal Processing Conference*, EUSIPCO '05.

[161] Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Spartan Books.

[162] Sakamoto, D., Komatsu, T., and Igarashi, T. (2013). Voice augmented manipulation: using paralinguistic information to manipulate mobile devices. In *Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services*, MobileHCI '13, pages 69–78, New York, NY, USA. ACM.

[163] San Agustin, J., Skovsgaard, H., Hansen, J. P., and Hansen, D. W. (2009). Low-cost gaze interaction: ready to deliver the promises. In *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, pages 4453–4458, New York, NY, USA. ACM.

[164] Schadle, I. (2004). Sibyl: AAC System Using NLP Techniques. In Miesenberger, K., Klaus, J., Zagler, W., and Burger, D., editors, *Computers Helping People with Special*

Needs, volume 3118 of *Lecture Notes in Computer Science*, pages 1009–1015. Springer Berlin Heidelberg.

[165] Scheirer, E. and Slaney, M. (1997). Construction and evaluation of a robust multifeature speech/music discriminator. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2 of *ICASSP '97*, pages 1331–1334.

[166] Scott MacKenzie, I. and Ashtiani, B. (2011). Blinkwrite: efficient text entry using eye blinks. *Univers. Access Inf. Soc.*, 10(1):69–80.

[167] Sears, A., Karat, C.-M., Oseitutu, K., Karimullah, A., and Feng, J. (2001). Productivity, satisfaction, and interaction strategies of individuals with spinal cord injuries and traditional users interacting with speech recognition software. *Universal Access in the Information Society*, 1(1):4–15.

[168] Shaffer, J. P. (1995). Multiple hypothesis testing. *Annual Review of Psychology*, 46(1):561–584.

[169] Shafran, I. and Rose, R. (2003). Robust speech detection and segmentation for real-time asr applications. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1 of *ICASSP '03*, pages 432–435.

[170] Shavelson, R. (1996). *Statistical Reasoning for the Behavioral Sciences*. Allyn and Bacon.

[171] Shieber, S. M. and Baker, E. (2003). Abbreviated text input. In *Proceedings of the 8th international conference on Intelligent user interfaces*, IUI '03, pages 293–296, New York, NY, USA. ACM.

[172] Sholes, C. L. (1896). Improvement in type-writing machines. US Patent No 207559.

[173] Sibert, L. E. and Jacob, R. J. K. (2000). Evaluation of eye gaze interaction. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, CHI '00, pages 281–288, New York, NY, USA. ACM.

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

[175] Smith, B. A. and Zhai, S. (2001). Optimised virtual keyboards with and without alphabetical ordering: A novice user study. In Hirose, M., editor, *Proceedings of Interact 2001IFIP International Conference on HumanComputer Interaction*, INTERACT'01, pages 92–99. IOS Press.

[176] Solutions, T. (1998). The fitaly one-finger keyboard.

[177] Song, Y. C. (2010). Joystick text entry with word prediction for people with motor impairments. In *Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility*, ASSETS '10, pages 321–322, New York, NY, USA. ACM.

[178] Soukoreff, R. W. and MacKenzie, I. S. (2001). Measuring errors in text entry tasks: an application of the Levenshtein string distance statistic. In *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '01, pages 319–320, New York, NY, USA. ACM.

[179] Soukoreff, R. W. and MacKenzie, I. S. (2003). Metrics for text entry research: an evaluation of msd and kspc, and a new unified error metric. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 113–120, New York, NY, USA. ACM.

[180] Špakov, O. and Majaranta, P. (2008). Scrollable keyboards for eye typing. In *Proceedings of the 4th Conference on Communication by Gaze Interaction*, COGAIN '08, pages 63–66.

[181] Sporka, A., Kurniawan, S., and Slavk, P. (2006a). Non-speech operated emulation of keyboard. In Clarkson, J., Langdon, P., and Robinson, P., editors, *Designing Accessible Technology*, pages 145–154. Springer London.

[182] Sporka, A. J. (2008). *Non-speech Sounds for User Interface Control*. PhD thesis, CTU in Prague.

[183] Sporka, A. J. (2009). Pitch in non-verbal vocal input. *SIGACCESS Access. Comput.*, (94):9–16.

[184] Sporka, A. J., Kurniawan, S. H., Mahmud, M., and Slavík, P. (2006b). Non-speech input and speech recognition for real-time control of computer games. In *ASSETS '06: Proceedings of the 8th international ACM SIGACCESS conference on Computers and accessibility*, pages 213–220, New York, NY, USA. ACM.

[185] Sporka, A. J., Kurniawan, S. H., and Slavík, P. (2004). Whistling User Interface (U3I). In *8th ERCIM International Workshop "User Interfaces For All", LCNS 3196, Vienna*, pages 472–478. Springer-Verlag Berlin Heidelberg.

[186] Sporka, A. J., Kurniawan, S. H., and Slavík, P. (2007). Physical human factor in non-speech input. In *Prooceedings of the CHI 2007 Workshop Striking a C[h]ord: Vocal Interaction in Assistive Technologies, Games, and More*, pages 13–16.

[187] Sporka, A. J. and Slavík, P. (2008). Vocal control of a radio-controlled car. *SIGACCESS Access. Comput.*, (91):3–8.

[188] Sporka, A. J., Žikovský, P., and Slavík, P. (2006c). Explicative document reading controlled by non-speech audio gestures. In Sojka, P., Kopecek, I., and Pala, K., editors,

Text, Speech and Dialogue, volume 4188 of *Lecture Notes in Computer Science*, pages 695–702. Springer Berlin / Heidelberg.

[189] Stevens, S. S., Volkmann, J., and Newman, E. B. (1937). A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190.

[190] Tanaka-Ishii, K. and Frank, I. (2005). Dit4dah: Predictive Pruning for Morse Code Text Entry. In Su, K.-Y., Tsujii, J., Lee, J.-H., and Kwong, O., editors, *Natural Language Processing IJCNLP 2004*, volume 3248 of *Lecture Notes in Computer Science*, pages 765–775. Springer Berlin Heidelberg.

[191] Tanaka-Ishii, K., Inutsuka, Y., and Takeichi, M. (2002). Entering text with a four-button device. In *Proceedings of the 19th international conference on Computational linguistics - Volume 1*, COLING '02, pages 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

[192] Teahan, W. (1995). Probability estimation for PPM. In *In Proceedings NZCSRSC'95. Available from http://www.cs.waikato.ac.nz/wjt.*

[193] Thacher, H. C. (1949). The BINAC: A product of the Eckert-Mauchly Computer Corp. Eckert-Mauchly Computer Corp.

[194] Thurstone, L. L. (1927). A law of comparative judgments. *Psychological Review*, 34:273–286.

[195] Tindale, A., Kapur, A., and Fujinaga, I. (2004). Towards Timbre Recognition of Percussive Sounds. In *Proceedings of International Computer Music Conference*, ICMC 2004. University of Michigan.

[196] Tregoubov, M. and Birbaumer, N. (2005). On the building of binary spelling interfaces for augmentative communication. *Biomedical Engineering, IEEE Transactions on*, 52(2):300–305.

[197] Trewin, S. and Arnott, J. (2007). Text entry when movement is impaired. In MacKenzie, I. S. and K.Tanaka-Ishii, editors, *Text Entry Systems: Mobility, Accessibility, Universality*, pages 289–304. Morgan Kaufmann.

[198] Tu, J., Tao, H., and Huang, T. (2007). Face as mouse through visual face tracking. *Computer Vision and Image Understanding*, 108(12):35–40.

[199] Tuisku, O., Majaranta, P., Isokoski, P., and 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 & applications*, ETRA '08, pages 19–26, New York, NY, USA. ACM.

[200] Tuisku, O., Surakka, V., Rantanen, V., Vanhala, T., and Lekkala, J. (2013). Text entry by gazing and smiling. *Advances in Human-Computer Interaction*, 2013.

[201] Urbina, M. H. and Huckauf, A. (2010). Alternatives to single character entry and dwell time selection on eye typing. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, ETRA '10, pages 315–322, New York, NY, USA. ACM.

[202] Ústav Českého národního korpusu FF UK (2010). Český národní korpus: Srovnávací frekvenční seznamy. http://ucnk.ff.cuni.cz/srovnani10.php Retrieved 6 January 2011.

[203] Vertanen, K. and Kristensson, P. O. (2011a). The imagination of crowds: conversational aac language modeling using crowdsourcing and large data sources. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 700–711, Stroudsburg, PA, USA. Association for Computational Linguistics.

[204] Vertanen, K. and Kristensson, P. O. (2011b). A versatile dataset for text entry evaluations based on genuine mobile emails. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, New York, NY, USA. ACM.

[205] Vertanen, K. and MacKay, D. J. (2010). Speech dasher: fast writing using speech and gaze. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 595–598, New York, NY, USA. ACM.

[206] Vigouroux, N., Vella, F., Truillet, P., and Raynal, M. (2004). Evaluation od AAC for Text Input by Two Groups of Subjects: Able-bodied Subjects and Disabled Motor Subjects. In *Adjunct Proceedings of the 8th ERCIM Workshop User Interfaces For All*, UI4ALL 04.

[207] Žibert, J., Pavesić, N., and Mihelič, F. (2006). Speech/non-speech segmentation based on phoneme recognition features. *EURASIP J. Appl. Signal Process.*, 2006:47–47.

[208] Waller, S. D., Williams, E. Y., Langdon, P. M., and Clarkson, P. J. (2010). Quantifying exclusion for tasks related to product interaction. In Langdon, P. M., Clarkson, P. J., and Robinson, P., editors, *Designing Inclusive Interactions*, pages 57–68. Springer London.

[209] Wandmacher, T., Antoine, J.-Y., Poirier, F., and Départe, J.-P. (2008). Sibylle, an assistive communication system adapting to the context and its user. *ACM Trans. Access. Comput.*, 1(1):6:1–6:30.

[210] Wang, C., Guan, C., and Zhang, H. (2005). P300 brain-computer interface design for communication and control applications. In *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, pages 5400–5403.

[211] Ward, D. J., Blackwell, A. F., and MacKay, D. J. C. (2000). Dasher – a data entry interface using continuous gestures and language models. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, UIST '00, pages 129–137, New York, NY, USA. ACM.

[212] Watts, R. and Robinson, P. (1999). Controlling computers by whistling. In *Proceedings of Eurographics UK*.

[213] West, L. J. (1998). The standard and dvorak keyboards revisited: Direct measures of speed. Technical report, Santa Fe Institute. http://samoa.santafe.edu/media/workingpapers/98-05-041.pdf.

[214] Wilson, A. D. and Agrawala, M. (2006). Text entry using a dual joystick game controller. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 475–478, New York, NY, USA. ACM.

[215] Witten, I. H. and Bell, T. (1991). The zero-frequency problem: estimating the probabilities of novel events in adaptive text compression. *Information Theory, IEEE Transactions on*, 37(4):1085–1094.

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

[217] Wobbrock, J. and Myers, B. (2006). Trackball text entry for people with motor impairments. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 479–488, New York, NY, USA. ACM.

[218] Wobbrock, J. O., Myers, B. A., and Aung, H. H. (2004). Writing with a joystick: a comparison of date stamp, selection keyboard, and EdgeWrite. In *Proceedings of Graphics Interface 2004*, GI '04, pages 1–8, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. Canadian Human-Computer Communications Society.

[219] Wobbrock, J. O., Myers, B. A., and Kembel, J. A. (2003). EdgeWrite: a stylus-based text entry method designed for high accuracy and stability of motion. In *Proceedings of the 16th annual ACM symposium on User interface software and technology*, UIST '03, pages 61–70, New York, NY, USA. ACM.

[220] Wobbrock, J. O., Rubinstein, J., Sawyer, M. W., and Duchowski, A. T. (2008). Longitudinal evaluation of discrete consecutive gaze gestures for text entry. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, ETRA '08, pages 11–18, New York, NY, USA. ACM.

[221] Won, S. Y., Lee, D.-I., and Smith, J. (2007). Humming control interface for handheld devices. In *Proceedings of the 9th international ACM SIGACCESS conference on Computers and accessibility*, Assets '07, pages 259–260, New York, NY, USA. ACM.

[222] Yin, P.-Y. and Su, E.-P. (2011). Optimal character arrangement for ambiguous keyboards using a PSO-based algorithm. In *Natural Computation (ICNC), 2011 Seventh International Conference on*, volume 4, pages 2194–2198.

[223] Zhai, S., Hunter, M., and Smith, B. A. (2000). The metropolis keyboard - an exploration of quantitative techniques for virtual keyboard design. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, UIST '00, pages 119–128, New York, NY, USA. ACM.

[224] Zhai, S. and Kristensson, P.-O. (2003). Shorthand writing on stylus keyboard. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 97–104, New York, NY, USA. ACM.

[225] Zhao, X. A., Guestrin, E. D., Sayenko, D., Simpson, T., Gauthier, M., and Popovic, M. R. (2012). Typing with eye-gaze and tooth-clicks. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, ETRA '12, pages 341–344, New York, NY, USA. ACM.

[226] Zhu, Y., Shasha, D., and Zhao, X. (2003). Query by humming: in action with its technology revealed. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 675–675, New York, NY, USA. ACM.

Relevant publications of the author

Journal publications

[A1] **Ondřej Poláček**, Adam J. Sporka, Pavel Slavík. A Comparative Study of Pitch-Based Gestures in Non-Verbal Vocal Interaction. *Transactions on Systems, Man, and Cybernetics Part A*, vol. 42, no. 6, pp. 1567–1571, IEEE, 2012.
(Ratio: 45%, IF: 2.183)

[A2] Adam J. Sporka, **Ondřej Poláček**, Jan Havlík. Segmentation of Speech and Humming in Vocal Input. *Radioengineering*, vol. 21, no. 3, pp. 923–929, 2012.
(Ratio: 45%, IF: 0.687)

WoS publications

[A3] **Ondřej Poláček**, Zdeněk Míkovec. Understanding Formal Description of Pitch-Based Input. In *Proceedings of the Third international conference on Human-centred software engineering*, HCSE '10, Reykjavik, Iceland, pp. 190–197. Springer Berlin / Heidelberg, 2010.
(Ratio: 70%)

[A4] **Ondřej Poláček**, Zdeněk Míkovec, Adam J. Sporka, Pavel Slavík. Humsher: A Predictive Keyboard Operated by Humming. In *Proceedings of the 13th international ACM SIGACCESS conference on Computers and Accessibility*, ASSETS '11, Dundee, UK, pp. 75–82, ACM, New York, 2011.
(Ratio: 50%)

Scopus publications

[A5] **Ondřej Poláček**, Zdeněk Míkovec. Hands Free Mouse: Comparative Study on Mouse Clicks Controlled by Humming. In *Extended Abstracts of CHI 2010*, Atlanta, GA, pp. 3769–3774. ACM, New York, 2010.
(Ratio: 70%)

[A6] Adam J. Sporka, Torsten Felzer, Sri H. Kurniawan, **Ondřej Poláček**, Paul Haiduk, Ian Scott Mackenzie. CHANTI: Predictive Text Entry Using Non-verbal Vocal Input. In *Proceedings of the 2011 annual conference on Human factors in computing systems*, CHI '11, Vancouver, Canada, pp. 2463–2472, ACM, New York, 2011.
(Ratio: 15%)

[A7] **Ondřej Poláček**, Zdeněk Míkovec, Pavel Slavík. Predictive scanning keyboard operated by hissing. In *Proceedings of the 2nd IASTED International Conference on Assistive Technologies*, AT '12, Innsbruck, Austria, pp. 862–869, IASTED, 2012.
(Ratio: 70%)

[A8] **Ondřej Poláček**, Adam J. Sporka, Zdeněk Míkovec. Measuring Performance of a Predictive Keyboard Operated by Humming. In Miesenberger, K. and Karshmer, A. and Penaz, P. and Zagler, W. (eds) *Computers Helping People with Special Needs*, LNCS 7383, pp. 467–474, Springer Berlin / Heidelberg, 2012.
(Ratio: 70%)

[A9] Adam J. Sporka, **Ondřej Poláček**. Toward a Design of Word Processing Environment for People with Disabilities. In *Proceedings of the 14th international ACM SIGACCESS conference on Computers and Accessibility*, ASSETS '12, pp. 263–264, ACM, New York, 2012.
(Ratio: 30%)

[A10] **Ondřej Poláček**, Adam J. Sporka, and Brandon Butler. Improving the Methodology of Text Entry Experiments In *Proceedings of the IEEE 4th International Conference on Cognitive Infocommunications*, CogInfoCom '13, pp. 155-160, IEEE, 2013.
(Ratio: 33%)

Other publications

[A11] **Ondřej Poláček**, Zdeněk Míkovec, Adam J. Sporka, Pavel Slavík. New way of vocal interface design: Formal description of non-verbal vocal gestures. In *Proceedings of the 5th Cambridge Workshop on Universal Access and Assistive Technology*, CWUAAT '10, Cambridge, UK, pp. 137–144, Cambridge Press, 2010.
(Ratio: 70%).

[A12] Adam J. Sporka, **Ondřej Poláček**, Jan Murin. Three Scanning Methods for Text Cursor Manipulation. In Langdon, P. M. et al. (eds) *Inclusive Designing*, CWUAAT '14, pp. 27-37, Springer Berlin / Heidelberg, 2014.
(Ratio: 40%)

[A13] Antonín Pošusta, **Ondřej Poláček**, Adam J. Sporka, Tomáš Flek, and Jakub Otáhal. Character Input by Myoelectric Signals. Bulletin of Applied Mechanics, 2014.
(Ratio: 20%)

Remaining publications of the author

WoS publications

[A14] **Ondřej Poláček**, Ivo Malý, Ladislav Čmolík, Filip Hanzl, Zdeněk Míkovec, Pavel Slavík. Designing a user interface of interactive digital television for elderly people. In *Proceedings of Ageing 2012*, pp. 109–117, Psychiatrické centrum Praha, 2012.
(Ratio: 25%)

Scopus publications

[A15] **Ondřej Poláček**, Adam J. Sporka, Pavel Slavík. Music alphabet for low-resolution touch displays. In *Proceedings of the International Conference on Advances in Computer Entertainment Technology*, ACE '09 Athens, Greece, pp. 298–301, ACM, New York, 2009.
(Ratio: 50%)

[A16] Adam J. Sporka, **Ondřej Poláček**, Pavel Slavík. Comparison of Two Text Entry Methods on Interactive TV. In *Proceedings of EuroITV' 2012*, Berlin, Germany, pp. 49–52, ACM, New York, 2012.
(Ratio: 45%)

[A17] **Ondřej Poláček**, Martin Klíma, Adam J. Sporka, Pavel Žák, Michal Hradiš, Pavel Zemčík, Václav Procházka A Comparative Study on Distant Free-Hand Pointing. In *Proceedings of EuroITV' 2012*, Berlin, Germany, pp. 139–142, ACM, New York, 2012.
(Ratio: 60%)

[A18] Thomas Grill, **Ondřej Poláček**, Manfred Tscheligi. Methods towards API Usability: A Structural Analysis of Usability Problem Categories. In Winckler, M. and Forbrig, P. and Bernhaupt, R. (eds) *Human-Centered Software Engineering*, LNCS 7623, pp. 164–180, Springer Berlin / Heidelberg, 2012.
(Ratio: 33%)

[A19] Thomas Grill, **Ondřej Poláček**, Manfred Tscheligi. ConWIZ: A tool supporting contextual Wizard of Oz simulation. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, MUM '12, pp. 21:1–21:8, ACM, New York, 2012.
(Ratio: 33%)

[A20] **Ondřej Poláček**, Tomáš Pavlík, Adam J. Sporka. Efficiency of Multi-tap Text Entry Method on Interactive Television. In Habernal, I., and Matoušek, V. (eds) *Text, Speech, and Dialogue*, LNCS 8082, pp. 217–224, Springer Berlin / Heidelberg, 2013.
(Ratio: 60%)

[A21] **Ondřej Poláček**, Tomas Grill, Manfred Tscheligi. NoseTapping: What else can you do with your nose? In *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, MUM '13, pp. 32:1–32:9, ACM, New York, 2013.
(Ratio: 33%)

Other publications

[A22] Doris Zachhuber, Thomas Grill, **Ondřej Poláček**, Manfred Tscheligi. Contextual Wizard of Oz: A framework combining contextual rapid prototyping and the Wizard of Oz method. In Paterno, F., Ruyter, B., Markopoulos, P., Santoro, C., Loenen, E., Luyten, K. (eds) *Ambient Intelligence*, LNCS 7683, pp. 224–239, Springer Berlin / Heidelberg, 2012.
(Ratio: 25%)

[A23] **Ondřej Poláček**, Thomas Grill, Manfred Tscheligi. Towards a Navigation System for Blind People: A Wizard of Oz Study. *SIGACCESS Accessible Computing*, no. 104, pp. 12–29, ACM, New York, 2012.
(Ratio: 33%)

Citations

Citations of [A5]

[C1] S. K. Tang, W. C. Tseng, W. W. Luo, K. C. Chiu, S. T. Lin, Y. P. Liu. Virtual Mouse: A Low Cost Proximity-Based Gestural Pointing Device. In J. Jacko (ed) *Human-Computer Interaction. Interaction Techniques and Environments.*, LNCS 6762, pp. 491–499, Springer Berlin / Heidelberg, 2011.

Citations of [A6]

[C2] D. Sakamoto, T. Komatsu, T. Igarashi. Voice augmented manipulation: using paralinguistic information to manipulate mobile devices. In *Proceedings of the 15th international conference on Human-computer interaction with mobile devices and services*, MobileHCI '13, pp. 69–78, ACM, New York, 2013.

Citations of [A4]

[C3] A. Paepcke and S. Kairam. EchoTree: Engaged Conversation when Capabilities are Limited. *Technical report*, Stanford InfoLab, 2013.

Citations of [A17]

[C4] Z. R. Cochran. The bit dome: creating an immersive digital environment with a Kinect-based user interface. *Journal of Computing Sciences in Colleges*, vol. 29, no. 2, pp. 191–198, Consortium for Computing Sciences in Colleges, 2013.

[C5] G. Ren, and E. O'Neill. Freehand gestural text entry for interactive TV. In *Proceedings of the 11th European conference on Interactive TV and video*, EuroITV '13, pp. 121–130, ACM, New York, 2013.

[C6] D. Y. Hsiao, S. Cooper, C. Ballweber, and Z. Popovic. User Behavior Transformation through Dynamic Input Mappings. In *Proceedings of Foundations of Digital Games 2014*, 2014.

Citations of [A16]

[C7] G. Ren, E. O'Neill. Freehand gestural text entry for interactive TV. In *Proceedings of the 11th European conference on Interactive TV and video*, EuroITV '13, pp. 121–130, ACM, New York, 2013.

[C8] A. Barrero, D. Melendi, X.G. Paneda, R. Garcia, S. Cabrero. An empirical investigation into text input methods for interactive digital television applications. *International Journal of Human-Computer Interaction*, Taylor & Francis.

Citations of [A23]

[C9] H. Paredes, H. Fernandes, P. Martins, J. Barroso. Gathering the Users' Needs in the Development of Assistive Technology: A Blind Navigation System Use Case. In C. Stephanidis, M. Antona (eds) *Universal Access in Human-Computer Interaction. Applications and Services for Quality of Life*, LNCS 8011, pp. 79–88, Springer Berlin / Heidelberg, 2013.

[C10] E. Firmino and M. Teófilo. Enriquecendo a experiência de uso do piso tátil com audiodescrições providas por celular In *Proceedings of the 12th Brazilian Symposium on Human Factors in Computing Systems*, ICH '13, pp. 260–263, Brazilian Computer Society, 2013.

Citations of [A2]

[C11] M-D. Zbancioc, S.M. Feraru. The automatic segmentation of the vocal signal using predictive neural network. In *Proceedings of 2013 International Symposium on Signals, Circuits and Systems*, ISSCS '13, pp. 1–4, IEEE, 2013.

[C12] T. Ramalingam and P. Dhanalakshmi. Speech/music classification using wavelet based feature extraction techniques. *Journal of Computer Science*, vol. 10, no. 1, pp. 34–44, Science publications, 2014.

Citations of [A18]

[C13] M. Doedt. Service-Integration in Geschäftsprozessmanagementsystemen mit besonderem Fokus auf die Integration von ERP-Systemen unter Bercksichtigung des aktuellen Trends hin zum Cloud-Computing. PhD thesis, Technische Universität Dortmund, 2013.

[C14] B. Cassell, and T. Szepesi. Web APIs, Like Caterpillars, Are Temporary. *Technical report*, University of Waterloo, 2014.

A Formal Description of Pitch-Based Input

The research described in this appendix has been published in [A11] and [A3].

The formal description of NVVI is based on *context-free grammar* (CFG). For a designer, who would not be an expert in CFG, it would be rather complicated to design the gestures in a form of CFG rules as a deeper knowledge of CFG theory is needed. For this reason, a specific way of gesture description have been developed—Vocal Gesture Template (VGT) expression. A VGT expression has similar structure to regular expression, which is more intuitive and easier to process for the designers. A VGT expression describes one or more gesture templates, its expected pitch profile and length.

Data from the low-level recognizer are expected in a form of sequence of frames, where each frame is described by extracted features of sound such as pitch, volume, timbre, etc. Those frames are considered as input symbols and following symbols are distinguished:

- p (pitch frame) stands for a frame that contain valid pitch data
- s (silence frame) stands for silence.

A gesture instance can be described by a sequence of such input symbols, as the frames are being retrieved from the low-level recognizer. For example, symbol sequence "pppppppps" describes a sound signal where 8 frames with tone are followed by 1 silent frame. Each input symbol can be further qualified by its attributes, such as pitch of the tone within that frame, volume, etc. The recognition is a process in which a particular sequence of input symbols is matched to the VGT expressions. Syntax of all VGT expressions can be partially expressed by Extended BackusNaur Form (EBNF):

```
Expr = ExprPart | Expr ‘‘|’’ Expr
ExprPart = Term [Output] [Quant]
Term = s | p [number] [‘‘[’’ Condition ‘‘]’’] | ‘‘(’’Expr‘‘)’’
Output = ‘‘<’’ outname attribute ‘‘>’’
Quant = *[min;[max]]
```

An expression *Expr* consists of consecutive units (*ExprPart*) or expressions connected by disjunction operator | that are matched in parallel. A *Term* can be symbol p (pitch frame), s (silence frame) or a VGT expression *Expr* enclosed in parentheses. Each p can be numbered, thus the actual pitch value can be used it in subsequent parts of the expression. A *Condition* denoted by "[", "]" can be located after p and it determines, whether a pitch frame is matched. Output symbols are enclosed in brackets "<", ">" and they provide notifications that can be mapped to application actions. They consist of notification name (*outname*) its attributes. Quantification operator $*$ is used to define a period (*min* and *max* values), in which appropriate frames are accepted, as shown in examples below. We will demonstrate the formal description on an existing application - mouse pointer controlled by voice [185].

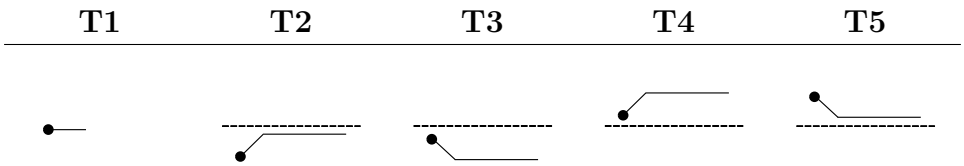


Figure A.1: Vocal gestures used to control mouse pointer. T1 - click, T2 - to the right, T3 - to the left, T4 - upwards, T5 - downwards.

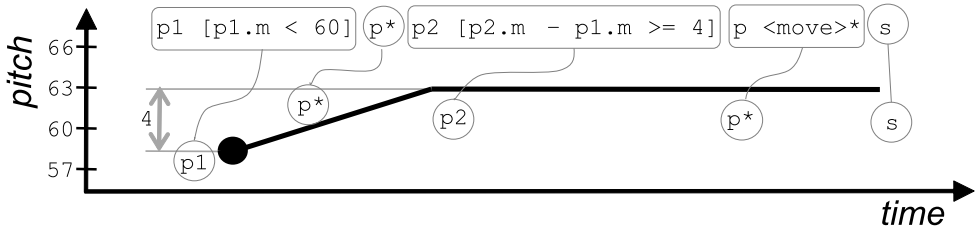


Figure A.2: Relation between graphical representation and VGT expression.

In the application, five vocal gestures are used as depicted in Figure A.1. Gesture T1 triggers a mouse click when the user produces a short tone. Gestures T2 and T3 drive the mouse cursor horizontally depending on tonal inflection (i.e. increase or decrease in a pitch) and gestures T4 and T5 drive the mouse cursor vertically. Vertical or horizontal direction of the cursor movement is determined by initial pitch, which is either lower or higher than a user-specified threshold pitch (see Figure A.1). Those vocal gestures can be easily defined by our VGT expressions as follows:

```
T1 = p*200;500 s <click>
T2 = p21 [p21.m < $TH] p* p22 [p22.m  p21.m > 8]
p23 <left p21.m p23.m>* s
T3 = p31 [p31.m < $TH] p* p32 [p31.m  p32.m > 8]
p33 <right p31.m p33.m>* s
T4 = p41 [p41.m >= $TH] p* p42 [p42.m  p41.m > 8]
p43 <up p41.m p43.m>* s
T5 = p51 [p51.m >= $TH] p* p52 [p51.m  p52.m > 8]
p53 <down p51.m p53.m>* s
```

Gesture template T1 defines short tone of any pitch profile that lasts from 200 to 500 ms. Quantification operator $*$ is used in regular expressions to match the preceding element zero or more times. In order to keep independence on frame rate, the operator $*$ in our approach defines period (*min*, *max*), in which appropriate frames are accepted. After accepting silence frame s, output *click* is triggered.

Gesture template T2 defines such instances, in which the first pitch is lower than a threshold pitch and which has a significant increase in a pitch profile. Relation between the expression T2 and its graphical form is shown in Figure A.2. It illustrates process of matching frame sequence and input symbols of the T2 expression. This process can be divided into four

parts (A-D):

A. In the first part, a pitch frame ($p21$) is matched only when its pitch attribute ($p21.m$) is lower than a constant $\$TH$ (threshold pitch TH depicted as dashed line). This is ensured by the condition $[p21.m < \$TH]$

B. Then all pitch frames ($p*$) are matched until difference between pitch attributes m of a current frame and the frame $p21$ is higher than 8 semitones (frame $p22$), which is defined by condition $[p22.mp21.m > 8]$

C. After satisfying the condition in step B, all pitch frames ($p23 < ... > *$) are matched and output symbol left is triggered with each matched frame. The symbol left has two attributes $p21.m$ and $p23.m$

D. Processing of the template T2 is finished, when a silence frame (s) is matched.

Key VGT expression features are explained in the following list:

1. Conditions are included in VGT expressions in order to solve the problem with wide variations of pitch values of gesture instances valid for one template. Various pitch restrictions such as "consider only tones lower than a threshold value" can be controlled by conditions as described above.

2. The m attribute is a note number logarithmically dependent on a tone frequency, which corresponds to human perception of a pitch. Other sound features such as volume or timbre can be easily added according to capabilities of the low-level recognizer.

3. Constants denoted by $ sign can be used to personalize VGT expressions according to user preferences.

4. Output symbols can be located anywhere in an expression and they can provide both event (T1) and continuous (T2-5) input channel.

The applicability of VGT expressions has been verified in prototypes used in chapters 6, 7, 8, and 9. An experiment was conducted by the author of this thesis with eight interaction designers to study their ability to comprehend the formal description [A3].

B Lists of Abbreviations and Acronyms

ANOVA	Analysis of variance
ASR	Automatic speech recognition
CFG	Context-free grammar
CHANTI	Vocally enhanced ambiguous non-standard text input
CPM	Characters per minute
CPS	Characters per second
EBNF	Extended BackusNaur form
EEG	Electroencephalography
EMG	Electromyography
ER	Error rate
FA	Friedreich ataxia
GPC	Gestures per characters
GUI	Graphical user interface
HCI	Human-computer interaction
HTML	Hypertext markup language
IAC	Important amplitude changes
ICT	Information and communications technology
ISO	International organization for standardization
IT	Information technology
KSPC	Keystrokes per character
KSR	Keystroke saving rate
LJC	Law of comparative judgment
MDITIM	Minimal device independent text input method
MFCC	Mel-frequency cepstral coefficients
MLP	Multilayer perceptron
mRMR	Minimum redundancy maximum relevance
MSD	Minimum string distance
NVVI	Non-verbal vocal input
NP	Nondeterministic polynomial time
PC	Personal computer
PPM	Prediction by partial matching
QANTI	Quick ambiguous non-standard text input
RMS	Root mean square
RSVP	Rapid serial visual presentation

SD	Standard deviation
SAK	Scanning ambiguous keyboard
SPC	Seconds per character
SPC	Scan steps per character
SPS	Scan steps per selection
TukeyHSD	Tukey's honest significant differences
T9	Text on 9 keys
VGT	Vocal gesture template
WPM	Words per minute