Chapter 14

New way of vocal interface design: Formal description of non-verbal vocal gestures

O. Poláček, Z. Míkovec, A.J. Sporka and P.Slavík

14.1 Introduction

Multimodal user interfaces become a common way of making the human-computer interaction more accessible by allowing the users to employ various modalities that suit best their capabilities. The acoustic input systems seem to be very promising, especially for users with motor disabilities, for whom they represent an additional or alternative input channel. Many researchers are devoted to the field of the automatic speech recognition. One of the main advantages of the speech-based interaction is the potential to reach very high level of intuitiveness, especially in transactional interactions (Cohen *et al.*, 2004). There is also an ongoing research in the field of non-verbal vocal interaction (NVVI), i.e. using other sounds than speech such as humming to control user interfaces. The NVVI can further exploit the potential of acoustic modality. The NVVI has already received a significant attention within the research community. It has proven it is a useful input modality for people with motor disabilities (Sporka, 2009; Harada *et al.*, 2006) as well as a voice training tool (Hämäläinen *et al.*, 2004). The NVVI input consists of non-verbal vocal gestures (NVVG) that are according to (Sporka *et al.*, 2006a) defined as short melodic patterns of defined pitch profile, length, volume, timbre, etc. We have developed a system that is capable of recognizing vocal gestures, which are explicitly defined by formal description. The system targets two user groups. The first one includes designers of NVVI application, for whom our system facilitates the definition of gestures by means of a formal description. Second user group are people with disabilities who use voice-based assistive tools.

Interfaces based on NVVG are less intuitive than speech based interfaces but can overcome some inherent problems of the speech recognition systems. The first problem is related to a real-time control of an application. The speech-based control can produce unacceptable time delays in the system reactions (Sporka *et al.*,

2006a). Related issue is the problem how to realize continuous control. The second problem of speech interaction is complicated provision of cross-cultural and language independent solutions.

The NVVI seem to solve all above mentioned problems. The recognition is simpler and thus more robust. It can be used for real-time control and can easily be designed as cross-cultural (Sporka *et al.*, 2006b). On the other hand the NVVG bring new problems that should be tackled: The designer has to start with modelling of the gestures, then assigns them to the application actions, and finally the gesture recognition must be solved. Unfortunately there is no general recognition system similar to automatic speech recognition (ASR).

The first problem is that the gestures must be designed in such a way, that it is easy to produce and remember them by the user. Users have different vocal capabilities (e.g., pitch range) and thus the personalization of the gestures is necessary to gain an acceptable individual usability level of the NVVG interaction.

The second problem is that the gestures must be recognizable by the recognition system. With increasing number of gestures (Sporka *et al.*, 2006c) it can happen, that some gestures will be accidentally designed in such a way, that either the system will never recognize them or the system will indicate recognition of more than one gesture at once. The identification of these interfering gestures is a non-trivial problem. We can see that the NVVG design process is a complex and multi-parametric task. A promising recognition method can be based on the hidden Markov models (HMM) or neural networks (Jin *et al.*, 2000). However, the use of HMM and neural networks makes it impossible to define the gestures explicitly and difficult to do changes to them (training on a set of real gestures is necessary). Moreover to solve the personalization of gestures (e.g., change of gesture length, pitch range) hierarchical HMM must be introduced (Chambers *et al.*, 2002). NVVG typically contain long-lasting constant parts, which cannot be properly recognized by HMM without introducing additional time duration triggers which can overcome freezing in a HMM state (Rabiner, 1989). This makes it even more difficult for the designer to analyze possible interference between gestures.

The third problem is nonexistence of an automatic recognition system for vocal gestures similar to ASR. Such a recognition system would allow the designer to model a gesture in more abstract way without necessity to implement the recognition system itself.

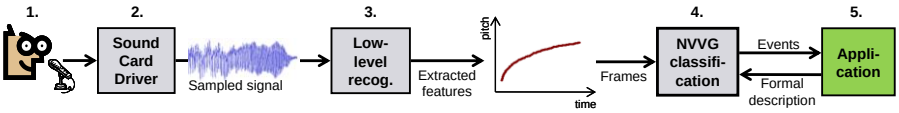

Figure 14.1. Sound recognition pipeline.

We introduce a formal description of NVVG based on *context-free grammar formalism* (CFG), that we have implemented in our own generic NVVG recognition system (see Fig. 14.1) and which solves the above mentioned problems. Our approach has the following features:

- allows the designer to design gestures in an explicit way, which helps the designer easily change and personalize gestures and solve interferences between them,
- separates the low-level part of gesture recognizer (analysis of the sampled audio signal) from the higher-level one, where the gestures can be defined in an abstract way (CFG based formal description),
- provides automatic gesture recognition based on designer-defined abstract way of gesture description (CFG based formal description).

The first three steps of the sound recognition pipeline (depicted in Fig. 14.1) are being frequently used in existing applications. They comprise of voice recording, signal sampling by sound card, and low-level extraction of voice features (pitch, volume, etc.). We implemented the fourth step – gesture classification, which is separated from both, the low-level recognizer and the application. The extracted features (third step) are passed to NVVG classifier in a form of frames which are short slots up to 100ms. The frames represent input symbols for evaluation of the CFG formal description.

14.2 Non-Verbal Vocal Gestures

The way the gestures are interpreted is defined by the application to which they are assigned. For example, the pitch of the tone may be coupled with a position of a game character on the screen (Hämäläinen *et al.*, 2004), or a particular melody may trigger an operating system macro that is assigned to this melody (Watts *et al.*, 1999). These two examples also show two different ways how the gestures can be interpreted:

- *continuous input channel***:** Specific acoustic feature of the gesture is constantly tracked and its value controls a user interface element, e.g., the Whistling User Interface (Sporka *et al.*, 2004), Vocal Joystick (Harada *et al.*, 2006) or Voice controlled plotter (Al-Hashimi, 2006),
- *event input channel***:** The gesture is recognized as a whole and an action that corresponds to this gesture, is carried out, e.g., whistling-controlled UNIX OS (Watts *et al.*, 1999) or acoustic emulation of keyboard control (Sporka *et al.*, 2006c).

Typical NVVI gestures can be described verbally such as “hold a tone and increase pitch or decrease pitch to move the character on screen”. Therefore particular gesture instances (the gestures as produced by the user) can vary in pitch, length, and other parameters. We must introduce some kind of gesture template, which will represent all gesture instances of given class. Our approach is to use the context-free grammars as a theoretical basis for such a description. The gesture template should be an approximate description of behaviour of sound properties of all valid gesture instances.

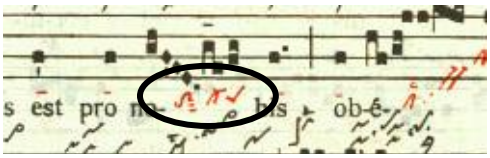

Figure 14.2. Cheironomic neumes, 9th century AD.

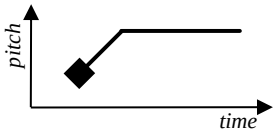

Figure 14.3. A Gesture template.

An interesting inspiration for graphical description of vocal gestures can be found in the music notation from the 9th century (Graduale triplex Relié, 1979). The system of *cheironomic neumes* (see an example in Fig. 14.2) provided relative description of pitch and length of particular words and syllables in the chant. Such a vague notation is very similar to graphical description of vocal gestures used in previous work (Sporka, 2009).

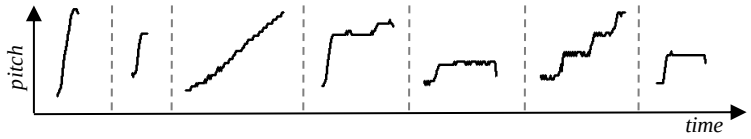

Figure 14.4. Gesture instances.

For example a gesture template (see Fig. 14.3) is used to describe various gesture instances (see Fig 14.4). The first impression is that those instances do not follow one particular template, however, we can observe that all instances has one common feature – the difference between initial and at least one subsequent pitch is greater than certain threshold. This approximate description has to be taken into consideration in our proposed formal description of gesture templates.

14.3 Formal description of gesture templates

Our proposed formal description is based on *context-free grammar* (CFG). We were aware that 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 we have developed specific way of gesture description – 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 (see Fig. 14.1), where each frame is described by extracted features of sound such as pitch, volume, timbre, etc. We consider those frames as input symbols and we distinguish following symbols:

- 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. E.g. 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 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 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 (Sporka *et al.*, 2004).

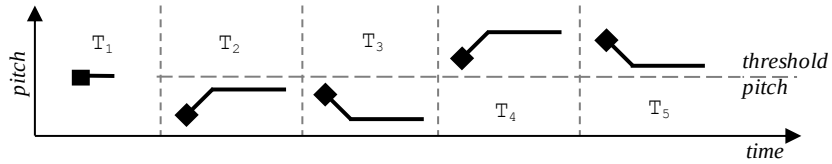

Figure 14.5. Vocal gestures used to control mouse pointer. T1 – click, T2 – to the right, T3 – to the left, T4 – upwards, T5 – downwards.

In the application five vocal gestures are used as depicted in Fig. 14.5. 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 Fig. 14.5). 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 independency 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.

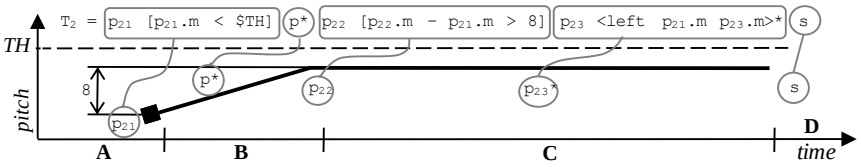

Figure 14.6. Relation between graphical representation and VGT expression.

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 Fig. 14.6. 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.m – p21.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 (see Fig. 14.1)
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.

Concept of VGT expression has been verified in several user tests. VGT expressions has been successfully applied to gestures from existing systems such as acoustic control of mouse pointer (Sporka *et al.,* 2004), Tetris game (Sporka *et al.,* 2006a), document reading (Sporka *et al.,* 2006b) and keyboard emulation (Sporka *et al.,* 2006c) that makes use of 43 different gestures (what is a non-typical use-case for NVVI).

14.4 Conclusion and Future Work

In this paper we introduce a novel approach to explicit description of non-verbal vocal gestures (VGT expression) and implementation of a general recognition system, which recognizes vocal gestures described by VGT expressions. These expressions are based on context-free grammar formalism. VGT expression formalism was successfully applied to existing NVVI systems.

NVVI is a proven assistive technology and our system accelerates the process of building the NVVI applications. The system is targeted to both designers and users with disabilities who use the recognition system incorporated in the assistive application. The main feature of VGT expressions is the possibility to enable continuous UI control and to solve inaccuracy in the pronunciation of vocal gestures.

For the future there is a range of open questions. A VGT expression can contain semantic errors, which cause improper behaviour in gesture recognition. Therefore a debugging tool, which allows visual detection of semantic errors, is needed. Next, the designers will cope with the problem of overly constraining of the template definition. In the field of the personalization of a set of templates based on capabilities of particular user we can try to define rules for transformation of the whole set of templates to another set designed for different user group. There is also a possibility to combine a grammar for automatic speech recognition together with VGT expression. A tool for converting visual representation of a gesture to the formal description would be also very useful for the designer. Recognition accuracy remains an unanswered question, however, the accuracy seems to be highly dependent on quality of low-level recognizer.

14.5 Acknowledgements

This research has been partially supported by the MSMT under the research program MSM 6840770014 and EC funded project VitalMind (ICT-215387). Adam Sporka's research is currently being supported by European Commission Marie Currie Excellence Grant for the ADAMACH project (contract No. 022593).

14.6 References

Al-Hashimi S (2006) Blowtter: A voice-controlled plotter. In: 20th BCS HCI Group conference in co-operation with ACM on HCI Engage, vol. 2, England, pp 41-44.

Chambers GS, Venkatesh S, West GAW, Bui HH (2002) Hierarchical recognition of intentional human gestures for sports video annotation. In: 15th International Conference on Pattern recognition, IEEE, vol. 2, pp 1082-1085

Cohen MH, Giangola JP, Balogh J (2004) Voice User Interface Design. Addison Wesley Longman Publishing Co., Inc. USA

Graduale triplex Relié (1979) Editions de Solesmes 1979, CARDINE (Dom Euge`ne). ISBN 2-85274-044-3

Harada S, Landay JA, Malkin J, Li X, Bilmes JA (2006) The vocal joystick: evaluation of voice-based cursor control techniques. In: 8th international ACM SIGACCESS conference on Computers and accessibility, USA, pp 197-204

Hämäläinen P, Mäki-Patola T, Pulkki V, Airas M (2004) Musical computer games played by singing. In: 7th International Conference on Digital Audio Effects, pp 367-371.

Jin HK, Kye KK, Nadal CP, Suen CY (2000) A methodology of combining HMM and MLP classifiers for cursive word recognition. In: 15th International Conference on Pattern recognition, vol. 2, IEEE, Spain, pp 319-322

Rabiner LR (1989) A tutorial on hidden markov models and selected applications in speech recognition. In: Proceedings of the IEEE, vol. 77, issue 2, pp 257-286

Sporka AJ (2009) Non-speech Sounds for UI Control. PhD thesis, CTU Prague

Sporka AJ, Kurniawan SH, Mahmud M, Slavik P (2006a) Non-speech input and speech recognition for real-time control of computer games. In: 8th international ACM SIGACCESS Conference on Computers and Accessibility, ACM, pp 213-220

Sporka AJ, Zikovsky P, Slavik P (2006b) Explicative Document Reading Controlled by Non-speech Audio Gestures. In 9th International Conference on Text, Speech and Dialogue, LNAI 4188, Springer-Verlag, pp 695-702

Sporka AJ, Kurniawan SH, Slavik P (2006c) Non-speech operated emulation of keyboard. In: Designing Accessible Technology, Springer-Verlag, pp 145-154

Sporka AJ, Kurniawan SH, Slavik P (2004) Whistling user interface (u3i). In: 8th ERCIM International Workshop on UI For All, LCNS 3196, Springer-Verlag, pp 472-478

Watts RP, Robinson P (1999) Controlling computers by whistling. In: Proceedings of Eurographics, Cambridge, UK