Understanding Formal Description of Pitch-Based Input

Ondřej Poláček and Zdeněk Míkovec

Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo nam. 13, 12135 Prague 2, Czech Republic
{polacond,xmikovec}@fel.cvut.cz

Abstract. The pitch-based input (humming, whistling, singing) in acoustic modality has already been studied in several projects. There is also a formal description of the pitch-based input which can be used by designers to define user control of an application. However, as we discuss in this paper, the formal description can contain semantic errors. The aim of this paper is to validate the formal description with designers. We present a tool that is capable of visualizing vocal commands and detecting semantic errors automatically. We have conducted a user study that brings preliminary results on comprehension of the formal description by designers and ability to identify and remove syntactic errors.

Keywords: Non-verbal Vocal Interaction; Vocal Gesture; Formal Description; User Study

1 Introduction

The *Non-Verbal Vocal Interaction* (NVVI) can be described as a method of interaction, in which sounds, other than speech, are produced. There are several approaches described in the literature which include using pitch of a tone, length of a tone, volume, or vowels in order to control the user interfaces. The NVVI is an interaction method that has already received a significant focus within the research community. It has been used as an input modality for people with motor disabilities [7][3] as well as voice training tool [2]. It is a method that shares some similarities with *Automatic Speech Recognition* (ASR). However, when comparing both interaction styles, several differences are revealed. Several reports, including mouse emulation [1] or controlling real-time games [7], suggest that NVVI is better fitted to continuous control rather than ASR. NVVI is cross-cultural and language independent [8]. Unlike ASR, NVVI generally employs simple signal processing methods [3]. Due to NVVIs limited expressive capabilities, ASR is better at triggering commands, macros or shortcuts. NVVI should be considered as a complement to ASR rather than replacement.

To design an application controlled by speech a set of word patterns or grammar must be defined. This grammar will then allow the ASR to recognize a range of expected words used in utterances. Likewise, a designer can also use a similar formal method for pitch-based NVVI.

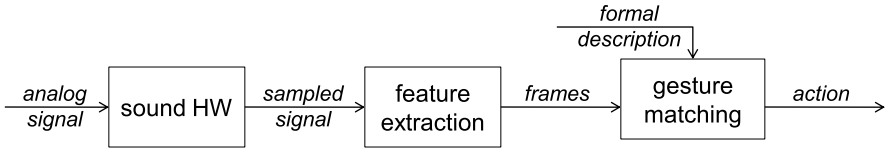

Fig. 1. NVVI signal processing pipeline

The signal processing pipeline for most pitch-based NVVI systems is depicted in Figure 1. Pitch is extracted from the sampled signal in a short discrete periods of time called frames. The typical duration of one frame is approximately 20 ms. The formal description of the NVVI and a stream of frames are then matched together, followed by generation of an appropriate action.

2 Formal description

When designing a set of voice gestures, the designer must describe an ideal pitch profile for each gesture. These ideal pitch profiles are then referred to as *gesture templates* and they are usually represented in graphic form as shown in Figure 2. However, the users are unable to produce an ideal pitch profile. The interpretation of gesture templates 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. Note that slightly different instances share the same semantics defined by the gesture template which is in this case an increasing tone. Once gesture templates are designed in a graphic form, they

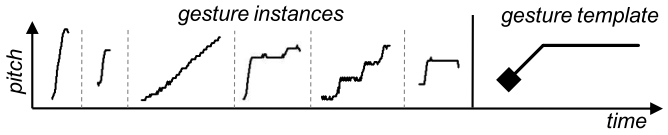

Fig. 2. Relationship between a gesture template and its instances

can be described by a *Voice Gesture Template* (VGT) expressions. Design of VGT expression is described in detail in [5]. These expressions are similar to regular expressions. They have two terminal symbols p and s that correspond to pitch and silence. They also use an operator * for repetition and operator | for the choice. However, there are several symbols with different meanings, for example brackets [ ] which are used for more sophisticated conditions and brackets <> which are used for output definitions to trigger an action. The use of VGT expressions is illustrated in Figure 3. The gesture template depicted in Figure 3 describes instances which start under midi note 60 and increase in pitch to more than 4 midi notes. Midi notes [4] are numerical representations of

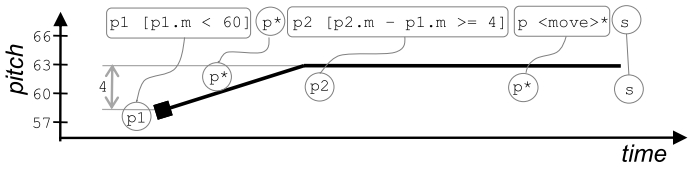

Fig. 3. VGT expression and its graphical representation of gesture template.

traditional notes in western music notation, for example, midi note number 60 corresponds to c'. The Figure 3 also illustrates the relation of VGT expression and the graphic representation of the gesture template. This process can be divided into four parts:

1. In the first part, the frame `p1` is matched to the expression when its pitch is under midi note 60. This is ensured by the condition `[p1.m < 60]` where the attribute `.m` is a midi note value of the frame `p1`;
2. Then all pitch frames `p*` are matched until the difference between the pitch of a current frame and the frame `p1` is higher than or equal to 4 midi notes (frame `p2`). This is ensured by the condition `[p2.m - p1.m >= 4]`;
3. After satisfying the condition in the 2nd step, all pitch frames `p <move>*` are matched and the output symbol `move` is triggered with each frame;
4. The processing of the template is completed, when a silent frame `s` is matched.

3 Semantic Errors

Semantic information, that describes pitch profiles of gesture templates, is encoded by a VGT expression. However, the description of gesture templates may be affected by semantic errors which cannot be detected while parsing the expression. A semantic error can also appear in a VGT expression when a new gesture template is added to the expression. The expression must be checked by tedious experimenting that involves user input to see if all templates are recognized correctly. Our research has identified two frequent types of semantic errors which cause improper behavior in gesture recognition – *ambiguous* and *unreachable* templates.

Two gesture templates are ambiguous if there is at least one gesture instance that satisfies both templates. The reason this error frequently occurs is due to an imprecise template description. In a real application there is typically a large number of instances fulfilling the condition of ambiguity. This semantic error is typically demonstrated by the generation of two or more output symbols in one frame.

The gesture template is unreachable when there is no instance matching the template. This can, for example, be caused by a condition that is always false, the template does not take into account human capabilities, or there is another gesture template that prevents the unreachable template from matching instances.

3.1 Semantic Error Detection

Detection of semantic errors, which are described above, requires analysis of gesture instances that can be generated by a VGT expression. We have implemented a tool which is capable of displaying possible gesture instances and automatically identifying semantic errors. It also allows deeper understanding of matching an instance to an expression by tracking its pitch profile. After generating all possible instances that match the expression, the tool checks if each instance belongs to just one template (ambiguity condition) and if each template has at least one instance (unreachability condition).

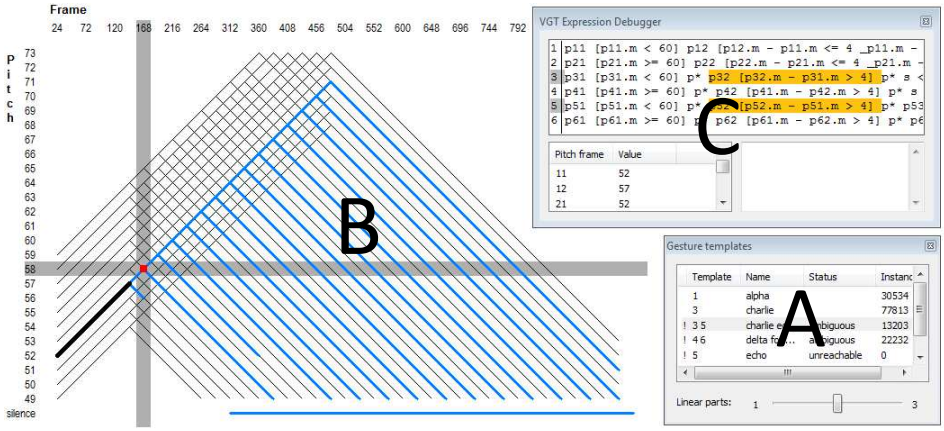

Fig. 4. Tool for vocal gestures visualization and semantic error detection.

The user interface of our tool is depicted in Figure 4. Part A of the Figure shows a dialog which contains a list of templates and number of instances. The dialog shows semantic errors within the VGT expression by displaying both ambiguous and unreachable templates (see the *Status* column). The user can display instances by selecting an appropriate row. When selecting a row with ambiguous templates, instances cause the ambiguity are displayed in part B.

Gesture instances are shown in the part B of Figure 4. The horizontal axis represents frames converted into timestamps in milliseconds and the vertical axis represents pitch using midi note numbers [4] starting with silence at the bottom. The black lines represent the generated gestures. When there are a lot of instances and their typical pitch profile is not visible, the user can display these instances and track them from the beginning to the end. When tracking an instance, the corresponding position of a VGT expression is highlighted in the VGT Expression Debugger (dialog in part C). Horizontal and vertical bars represent the current position, the bold line represents the part of an instance that has been already tracked and the blue lines show the further extending of a current instance.

The VGT expression is shown in dialog C. The current position of tracked instance is highlighted directly in the VGT expression by a yellow background, allowing the user to inspect how the instance is matched to its template. This is a very useful feature when inspecting instances that correspond to two or more ambiguous gestures, as the user can now clearly see the cause of the ambiguity. Current pitch values of numbered pitch frames are shown below the expression.

4 User study

The aim of the user study was to find out whether the designers could understand VGT expressions, and to demonstrate the usefulness of the tool described in the previous section. Eight designers were recruited to participate in the study. Each participant (mean age=29.6, SD=2.8) had some previous experience with NVVI – four of them knew the interaction method, three had used it at least once and one had previously designed an NVVI application. Seven of the participants considered themselves as interaction designers and the remaining one as a usability expert. All participant were familiar with regular expressions.

The participants were given approximately 20 minutes of training, which involved discussing the syntax of two VGT expression examples as well as semantic errors. The participants were asked to complete three tasks. In each task they were told to recognize the gesture templates in given VGT expression by describing them orally and sketching a graphic representation of each template. They were also asked to identify any semantic errors that may have been present in the expressions and to propose a solution for each. However, they were not told to write a new corrected expression due to limited time of each session. One session lasted approximately one hour. Participants were divided into two groups of four – Group A and B. *Group A* was allowed to use the tool described above, whereas *Group B* was not allowed to use any aid.

Task #1

In the first task participants were told to analyze the following VGT expression:

```
p1 p* (p2 [p2.m - p1.m > 4] p* s <alpha> |
       p3 [p2.m - p2.m > 8] p* s <bravo>)
```

The expression above describes the two templates as depicted in Figure 5a. The *alpha* template defines instances where pitch increases by 4 or more midi notes. The *bravo*'s instances have to increase by 8 midi notes. However, the *bravo* template is unreachable, as the condition in the *alpha* template is always matched earlier.

Group A (Use of tool): Each participant correctly understood the templates and discovered that the gesture *bravo* was unreachable. Two participants proposed a partially correct solution.

Group B: One participant misunderstood the *bravo* template and consequently could not see an error. The other participants miscategorized the error as ambiguous. Two participants proposed a partially correct solution.

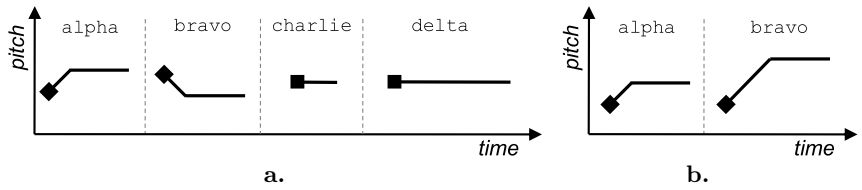

Fig. 5. a. Gesture templates in the task #1 **b.** Gesture templates in the task #2

Task #2

The second task contained gestures used in the *Tetris* game controlled by humming [7]. The participants were again told to analyze the VGT expression:

```
p1 p*
   (p2 [p2.m - p1.m > 4] p <alpha>* s |
    p3 [p1.m - p3.m > 4] p <bravo>* s) |
p*200;600 s <charlie> |
p*500; s <delta>
```

The expression above describes the templates depicted in Figure 5b. *Alpha* instances have to increase in pitch by 4 or more midi notes, whereas the *bravo* instances have to decrease by the same amount. *Charlie* instances are short tones of 200 to 600 ms and *delta* instances are all those that are longer than 500 ms. Two ambiguities are present in the expression. The first one is a time overlap in *charlie* and *delta* templates. The solution is to modify one of the limits. The second error is a pitch overlap between *alpha*, *bravo* and *charlie*, *delta* templates, due to the latter two not defining a pitch limit. The solution is to limit the pitch in *charlie*, *delta* templates to within ± 4 midi notes.

Group A (Use of tool): Each participant understood the presented templates. One participant incorrectly identified the gestures initially, but corrected their interpretation after using the tool. All four were also able to locate all errors and propose a correct solution for each error.

Group B: Unlike the three others, one participant was not able to describe *alpha* and *bravo* templates correctly. All four participants were able to find ambiguity between *charlie* and *delta*. The second error was found by three participants, who proposed a correct solutions for each of the errors.

Task #3

The most complex VGT expression was analyzed in the last task. The expression defines six of the eight templates used in keyboard controlled by humming [6].

```
p11 [p11.m< 60] p12 [p12.m-p11.m<=4 & p11.m-p12.m<=4]* s<alpha> |
p21 [p21.m>=60] p22 [p22.m-p21.m<=4 & p21.m-p22.m<=4]* s<bravo> |
p31 [p31.m< 60] p* p32 [p32.m-p31.m>4] p* s<charlie> |
p41 [p41.m>=60] p* p42 [p41.m-p42.m>4] p* s<delta> |
p51 [p51.m< 60] p* p52 [p52.m-p51.m>4] p* p53 [p53.m<=p51.m] p* s<echo> |
p61 [p61.m>=60] p* p62 [p61.m-p62.m>4] p* p63 [p63.m>=p61.m] p* s<foxtrot>
```

The six instances correspond to the following - 1. *alpha* to a straight low tone, 2. *bravo* to a straight high tone, 3. *charlie* to increasing tone by more than 4 midi notes, 4. *delta* to decreasing tone by more than 4 midi notes, 5. *echo* to a tone that increases by more than 4 midi notes and then decreases to at least its initial pitch and finally 6. *foxtrot* which is essentially *echo* vertically inverted. Ambiguities between *charlie* and *echo* and between *delta* and *foxtrot* are present due to the end pitch of *charlie* and *delta* templates not being limited.

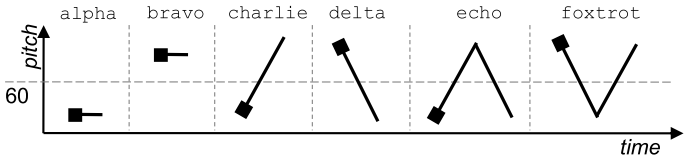

Fig. 6. Gesture templates in the task #3

Group A (Use of tool): One participant misunderstood *alpha* and *bravo* templates. Two participants incorrectly identified the templates initially, but corrected their interpretations after using the tool. Two participants thought there was an error present between *alpha* and *bravo*, but identified their mistake after using the tool. All participants located the error and three of them were able to propose correct removal solution.
Group B: Two participants incorrectly identified *alpha* and *bravo* templates as unreachable and were thus unable to sketch them. The other two participant incorrectly identified the templates as ambiguous. However, the other templates were understood by all participants, who were also able to identify the ambiguities and propose correct solutions.

5 Discussion

Using VGT expressions accelerates the process of building an NVVI application, as the matching algorithm no longer needs to be hard coded. The question, that is raised though, is whether designers are able to understand these VGT expressions. In most cases, participants from both groups correctly identified templates directly from VGT expression, which supported our assumption that VGT expressions can be understood by most designers. From total of 48 gestures that were examined in one group, there was two errors in the group A (use of tool) and seven error in the group B. What was slightly surprising was that participants from group A primarily relied on their own judgement rather than on the provided tool. However, they did use the tool from time to time to visually confirm their opinion or when they were unsure of the answer. In these situations the tool helped them to correctly understand the given templates and consequently to succeed in fulfilling the tasks. Thanks to the tool, participants from the group A also had no difficulty in detecting semantic errors. Although

the participants from the group B were not as successful as group A, they were still able to locate a significant number of error occurrences. It seems that the use of the tool results in better understanding of VGT expressions and minimizes the overlooking of semantic errors. However, a further quantitative study is needed in order to support this hypothesis.

6 Conclusion

This paper discusses the formal description of pitch-based vocal input, used during the design process of NVVI applications. We have created a tool for automatic error detection and visualization of the formal description. Our research was focused on the comprehension of the formal description by designers and their ability to detect possible semantic errors with and without using the tool. Their ability to comprehend the formal description and to detect semantic errors was validated in a user study by eight interaction designers. Designers who used the tool were more successful in understanding the formal description. Further research concerning these results will be conducted in the future, including a comparative quantitative study to prove the efficiency of the gesture visualization tool.

Acknowledgments. This research has been partially supported by the MSMT research program MSM 6840770014 and the VitalMind project (IST-215387).

References

1. Harada, S., Landay, J.A., Malkin, J., Li, X., Bilmes, J.A.: The vocal joystick: evaluation of voice-based cursor control techniques. In: Proceedings of ASSETS'06, pp 197-204. ACM Press (2006)
2. Hämäläinen, P., Mäki-Patola, T., Pulkki, V., Airas, M.: Musical computer games played by singing. In: 7th International Conference on Digital Audio Effects, pp 367-371. (2004)
3. Igarashi, T., Hughes, J.: Voice as sound: using non-verbal voice input for interactive control. In: Proceedings of UIST'01, pp. 155–156. ACM Press (2001)
4. MIDI Manufacturers Association: Complete MIDI 1.0 Detailed Specification v96.1 (2nd ed.), `http://www.midi.org/techspecs/midispec.php` (2001)
5. Poláček, O., Míkovec, Z., Sporka, A.J., Slavík, P.: New way of vocal interface design: Formal description of non-verbal vocal gestures. In: Proceedings of the CWUAAT'10, pp. 137–144. Cambridge Press, UK (2010)
6. Sporka, A.J., Kurniawan, M., Slavík, P.: Non-speech Operated Emulation of Keyboard. IIn: Designing Accessible Technology, ISBN 1-84628-364-7, pp. 145–154. Springer, Heidelberg (2006)
7. Sporka, A.J., Kurniawan, S.H., Mahmud, M., Slavík, P.: Non-speech Input vs Speech Recognition: Real-time Control of Computer Games. In: Proceedings of ASSETS'06, pp. 213–220. ACM Press (2006)
8. Sporka, A.J., Žikovský, P., Slavík, P.: Explicative Document Reading Controlled by Non-speech Audio Gestures. In Text, Speech and Dialogue, 9th International Conference, TSD 2006, LNAI 4188, pp. 695-702. Springer, Heidelberg (2006)