VIVO - VISUALIZING HARMONIC PROGRESSIONS AND VOICE-LEADING IN PWGL

Mika Kuuskankare
Sibelius Academy
CMT

Mikael Laurson
Sibelius Academy
CMT

ABSTRACT

This paper describes a novel tool called VIVO (VIsual VOice-leading) that allows to visually define harmonic progressions and voice-leading rules. VIVO comprises of a compiler and a collection of specialized visualization devices. VIVO takes advantage of several music related applications collected under the umbrella of PWGL (PWGL is a free cross-platform visual programming language for music and sound related applications). Our music notation application–Expressive Notation Package or ENP–is used here to build the user-interface used to visually define harmony and voice-leading rules. These visualizations are converted to textual rules by the VIVO compiler. Finally, our rule-based compositional system, PWGLConstraints, is used generate the final musical output using these rules.

1 BACKGROUND

The musical problems that are interesting in terms of musical constraints programming are typically very demanding. Here, harmony, melody, voice-leading, and counterpoint provide challenges not only in an aesthetic sense but also in terms of a formal definition. Defining textually rules that solve a certain compositional or music analytical problem is a time consuming task and requires a lot of both musical and programming expertise.

VIVO allows to visually define rudimentary rules of counterpoint. The approach presented here resembles the traditional teaching situation where a teacher or a textbook gives out examples of correct use of harmonic progressions or voice-leading. Using music notation as a framework for constructing the VIVO user-interface provides several advantages: (1) it is easy to write the rules, i.e., the user sees the exact musical context the rule is applied to; (2) it is possible to verify the correctness of the data by ’listening’ to the rules; (3) it is straightforward to edit, add and remove data; and (4) the user can easily edit the rule set and make subsets of it.

Our environment for computer assisted composition, PWGL ([3]; `http://www.siba.fi/pwgl/`), is used to implement VIVO. PWGL offers a unique combination of graphical and textual programming tools. The two of which are of primary importance in terms of VIVO are PWGLConstraints [4] and ENP [2].

© 2007 Austrian Computer Society (OCG).

VIVO uses an ENP score as a user-interface component. By entering musical material into the score makes it possible to define given aspects of harmony and voice-leading. The visual representation is then fed to the VIVO compiler which in turn generates textual rules that are suitable for PWGLConstraints. The final output generated by PWGLConstraints can then be shown in common music notation using ENP.

There are also other rule-based systems that have been used to solve musical constraint satisfaction problems, e.g., Situation, Arno, OMClouds, Strasheela (see [1] for more information) and the choral harmonizer system by Kemal Ebcioglu. Furthermore, integrating visual tools in constraints programming has been studied, for example, in [5]. However, using music notation to define rules of counterpoint, in the way the VIVO does, is unique and cannot be found elsewhere.

This paper gives a brief overview of some of the currently available VIVO tools.

2 THE VIVO TOOLS

2.1 Common Harmonic Progressions

Figure 1 gives an example of a simplified harmonic progression defined with the help of VIVO. There are two cases, marked as $\boxed{1}$ and $\boxed{2}$ in the example. The idea here is to give examples–written in music notation–of acceptable harmonic progressions. The setting or the inversion of the chords is not important.

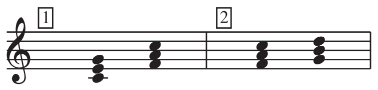

Figure 1. A simplified harmonic progression database.

The relationships between two adjacent chords in VIVO are relative it is possible that harmonic sequences, other than the ones explicitly indicated in the database, can be formed. For example, using the database given in Figure 1, it is possible to form among others the following sequence: I-V/V, V/V-V, V-V/II. This scheme actually allows VIVO to modulate from one harmonic region to another.

2.2 Common Voice-Leading Cases

2.2.1 Suspended Chords

There are certain graphical devices that can be used to define aspects of the voice-leading in more detail. Figure 2 shows how preparing and resolving a suspended harmony (e.g., I^{4-3}) is defined using VIVO. The horizontal lines connecting two adjacent notes are used to constrain the movement of a given member of the harmony. In Figure 2, $\boxed{1}$ defines how the suspended harmony is prepared and $\boxed{2}$ defines how it is resolved.

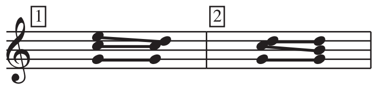

Figure 2. A VIVO database defining a suspended harmony and its resolution.

This database would produce an uninterrupted chain of chords cycling the pattern *prepared-suspended-resolved*. Figure 3 gives an example of a score, generated by VIVO, using the above database.

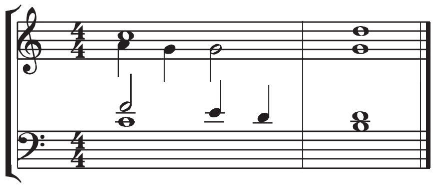

Figure 3. A four-voiced chorale fragment containing a chain of suspensions.

2.2.2 Neapolitan chord

As can be seen in Figure 3, the suspension database given above does not assume any particular inversion of the chords (or setting for that matter). However, in some cases–as in case of the Neapolitan Sixth Chord (N^6)–the inversion of the chord is of primary importance.Figure 4 gives a new visualization device (shown as a thick dashed line in the left hand staff) that is used to indicate the root note of a given chord.

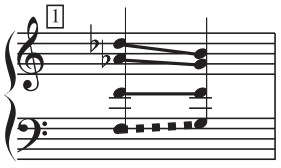

Figure 4. The Neapolitan chord and its idiosyncratic voice-leading and enharmonic content defined in VIVO.

3 DISCUSSION

VIVO is still more or less in the conceptual level. More work is needed to create a good set of visualization devices that are descriptive yet straightforward.

There are many open questions dealing with such fundamental voice-leading cases as parallel-, hidden-, similar-, oblique- or contrary-motion; voice-crossing or -overlapping; open- and close-positions; or cadences. It should also be investigated how to visualize key dependent issues such as the leading tone or harmonic regions, etc.

One of the most interesting applications in terms of MIR would be to let VIVO learn harmonic progressions from the repertoire, e.g., to analyze existing scores to produce the VIVO database.

Some of the issues enumerated above would probably be already possible to realize. Using and existing score to create a VIVO database from, for example, a Bach Chorale wouldn't require much extra work. Furthermore, open- and close-positions could very well be entered with the help of VIVO except that the VIVO rule compiler should be instructed to compile these as so called heuristic rules.

It is perhaps equally evident that not all of the features mentioned above are suitable to be represented visually. It is, however, possible to augment VIVO to cover a great deal of them.

4 ACKNOWLEDGMENTS

The work of Mikael Laurson and Mika Kuuskankare have been supported by the Academy of Finland (SA 105557 and SA 114116).

5 REFERENCES

[1] Anders, T., C. Anagnostopoulou, and M. Alcorn, "Strasheela: Design and Usage of a Music Composition Environment Based on the Oz Programming Model", *Multiparadigm Programming in Mozart/OZ: Second International Conference, MOZ 2004* (Roy, P. V., ed.), vol. LNCS 3389, Springer-Verlag, 2005.

[2] Kuuskankare, M. and M. Laurson, "Expressive Notation Package", *Computer Music Journal*, vol. 30, no. 4, pp. 67–79, 2006.

[3] Laurson, M. and M. Kuuskankare, "PWGL: A Novel Visual Language based on Common Lisp, CLOS and OpenGL", *Proceedings of International Computer Music Conference*, (Gothenburg, Sweden), pp. 142–145, 2002.

[4] Laurson, M. and M. Kuuskankare, "Extensible Constraint Syntax Through Score Accessors", *Journées d'Informatique Musicale*, (Paris, France), 2005.

[5] Schulte, C., "Oz Explorer: A visual constraint programming tool.", *Proceedings of the Fourteenth International Conference on Logic Programming*, The MIT Press, 1997.