EXTENDING VIVO AS A MIR SYSTEM

Mika Kuuskankare and Mikael Laurson

Sibelius Academy
Centre for Music and Technology
P.O.Box 86, FIN-02510, Finland
{mkuuskan,laurson}@siba.fi

ABSTRACT

In this paper we extend the VIVO system with MIR functionality that allows us to extract the VIVO rule database from a symbolic score. As a reference material we use a collection of selected Chorale harmonizations by J.S.Bach. Our extended system takes symbolic scores as input and generates a harmonic progression database as output. The VIVO and the new extended system are realized inside the PWGL environment. Complete working patches are given as examples. As a proof of concept the database-extracted from the repertoire-is then used in the VIVO to re-harmonize a short melody.

1. INTRODUCTION

Several different techniques have bee applied in producing chorale harmonizations automatically. One of the most known example is arguably the rule-based choral harmonizer system by Kemal Ebcioglu [1]. Schottstaed [2] takes rules from Fux [3] to produce species counterpoint. [4], in turn, applies genetic programming techniques to harmonization. Furthermore, [5] uses neural networks and [6] Markov models to generate harmonic structures. An overview of constraint-based harmonization systems is given in [7] and a comparison of constraint-based systems and genetic algorithms can be found in [8].

VIVO [9] is a system that can be used to visually create harmonization rules. It uses ENP [10] as the primary user-interface component. Rules are defined by giving examples in standard western music notation. VIVO includes a set of special visualization devices that can be inserted in music notation to define aspects of voice-leading, for example. The visual rules are translated into textual rules by the VIVO compiler. The advantages of VIVO are that it allows us: (1) to represent the harmonic progressions and voice-leading rules visually, (2) to represent the results in standard music notation, (3) to mix textual representation of rules with the visual ones, and (4) to transparently mix heuristics and counterpoint rules on top of the existing ones.

Our previous work [9] demonstrated how the system can be used to prepare some traditional voice-leading assignments, such as, suspensions, four-six progressions ($V\frac{6}{4}$–V), etc. While it is the purpose of VIVO to let the user to define the rules by hand it is also possible to create the whole rule database algorithmically.

In this paper our objective is to let VIVO learn harmonic progressions from the repertoire, i.e., to analyze existing scores to produce the visual VIVO rule database. We augment the VIVO system with MIR functionality using an object-oriented scripting language that allows us to extract the VIVO rule database from a collection of symbolic scores. One of the benefits of the new MIR functionality is that it is possible to extract a rich set of musical information using a sophisticated and object-oriented scripting language. Owing to it's clean and simple syntax it is also straightforward to convert these findings to generative rules (and also vice versa). All the tools described in this paper are realized inside PWGL [11].

In the rest of this paper we first discuss and describe the retrieval apparatus used in extracting the rule database from existing repertoire. In the remainder of the paper we use this database in conjunction with the original VIVO system to generate sample harmonization of a choral melody.

2. THE RETRIEVAL APPARATUS

In the center of our retrieval system is the rule-based scripting language ENP-Script [12]. ENP-Script can be used to produce side-effects to or to extract information from an ENP score. The syntax of the language is not presented here, instead the reader is advised to study, for example [13]. Furthermore, recent developments in the syntax are covered in detail in [14].

In this paper we use ENP-script to define a collection of scripts that extract interesting properties from ENP scores. The properties we are primarily interested in are melodic movement, voicing, harmonic progression, and voice-leading.

Currently, the rules fall into two main categories: (1) rules that define harmonic progression, and (2) rules that define voice-leading.

In the earlier system described in [9], it was not possible to define progressions that are key dependent. Here, the VIVO system has been augmented so that when a key signature is given in the database score, VIVO interprets the individual harmonic rules as absolute instead of relative. This allows us to define specific functions (e.g., I–IV–V) inside one key. In case of chorale harmonization this is an important feature as the harmonic regions seldom move very far from the key center.

Simplicity in mind, we use a small collection of hand-picked Chorale harmonizations by J.S.Bach as a starting point. The BWV 248.5, 244.62, 244.54, 244.44, and 244.17 of the chorale *Herzlich tut mich verlangen* were used. All the chorales were transposed in a-minor.

The main retrieval process consists of the following steps:

(1) Analyze all the scores for common tertian harmonies. This analysis is stored into a special variable called `:common-chords` (see the script given below).

(2) Run the retrieval script on all the scores to collect all the two consecutive sounding harmonies and record their relation to each other. Here we use the :common-chords property to distinguish between cases that contain passing tones from the ones that do not.

2.1. The Retrieval Patch

Figure 1 shows the 'retrieval' patch. The left part of the patch imports a collection of MIDI files in a loop and converts them

into ENP scores. The scores are not quantized here. On the right, there is a box called 'Multi-Score-Editor' displaying one of the imported scores. The Multi-Score-Editor can contain an arbitrary number of scores one of which is displayed at a time. The rest of the patch (below Multi-Score-Editor) deals with the retrieval of the properties.

2.2. The Retrieval Script

Next, we discuss the ENP-Script for retrieving harmonic progressions and voice-leading (see Figure 2).

It is important to realize that the script presented here is based on a special pattern-matching language that is used to extract relevant objects from the score. In the first line of the script (a) we define the pattern that we are interested in. Here, we want to record the relation between two consecutive harmonies in the score, thus the pattern `(* ?source-chord ?target-chord :harmony`. The keyword `:harmony` indicates to the script that in this case we are interested in the 'harmony' property of the score. Each of the consecutive harmonies are in turn bound to the two variables `?source-chord` and `?target-chord`.

The scripting engine is constructed so that it goes through all the partial harmonies (i.e., constructing every harmony note by note from bottom up).Thus, it is necessary to tell the system that in this case we are only dealing with complete harmonies. See the keyword `:complete?` used in the condition found in (c). The rest of the script is executed only when both of the harmonies contain all the sounding notes.

The rest of the script deals with assigning the current harmonic situation into one of the rule categories described above. The simple harmonic progression case is handled in (d-m) and the more elaborate one, in turn, in (o-z).

The former case is quite straightforward. When both of the harmonies are simple tertian they are just pushed into a variable called `:vivo-pre-analysis` (see h-m). This becomes then one of the harmonic progression rules in VIVO.

The latter case, however, requires some extra work. Here, the assumption is that one or both of the harmonies contain pitches that are the result of passing tone texture, for example. To assure correct voice-leading, in this case, we constrain the movement of the individual voices. In VIVO, this was done visually by inserting certain expressions in the notation to denote how the voices should move. Now, we simulate this algorithmically by collecting the notes in each of the harmonies (o-r), looping them through (s-z) and inserting an appropriate 'voice-leading' expression between the two melodic notes in each of the voices (v-x). Finally, the new construction is pushed in the list of VIVO rules (y-z).

Figure 3 illustrates how the retrieval script functions. Cases with two consecutive tertian chords are marked with rectangles vertically covering the notes of the harmony (see the rst and second measure). Cases where one (or both) of the chords is not a common tertian chord are in turn marked with lines indicating the passing tone movement (see, for example, the third and fourth measure). Note, that visualization is automatically generated. Here, we use a slightly modified retrieval script that instead of building the database inserts an appropriate expression in the score.

In this section we described the procedure behind retrieving harmonic progressions and voice-leading from ENP scores. In the next Section we attempt to validate the method by using the automatically extracted and generated visual VIVO rules to produce a re-harmonization of an existing melody.

```
a (* ?source-chord ?target-chord :harmony
b   (?if
c    (?incase (m ?target-chord :complete? t)
d      (cond ((and (member (sc-name (m ?source-chord))
e                          (pwgl-value :common-chords))
f                  (member (sc-name (m ?target-chord))
g                          (pwgl-value :common-chords)))
h             (pwgl-value :vivo-pre-analysis
i                         :push
j                         (list (duplicate-instance
k                                (m ?source-chord :object t))
l                               (duplicate-instance
m                                (m ?target-chord :object t)))))
n            (T
o             (let ((c1 (duplicate-instance
p                        (m ?source-chord :object t)))
q                   (c2 (duplicate-instance
r                        (m ?target-chord :object t))))
s               (loop for n1 in (sort c1 '< :key #'midi)
t                     for n2 in (sort c2 '< :key #'midi)
u                     do
v                     (insert-expression
w                      (list n1 n2)
x                      (make-instance 'voice-leading)))
y               (pwgl-value :vivo-pre-analysis
z                           :push (list c1 c2))))))))
```

Figure 2: The retrieval script realized with ENP-script language.

3. AN EXAMPLE RE-HARMONIZATION

Next, we use the harmonic progression database extracted from the example score to re-harmonize a short chorale melody. Figure 4 gives a complete VIVO patch consisting of (1) the visual rule database, (2) the textual rule generator, (3) the input score with a pre-constrained melody, (4) the output score with the re-harmonized chorale melody, (5) basic voice-leading rules (including rules excluding parallel movement, for example), (6) heuristic rules, and (7) the solver box.

When the patch is evaluated the visual rules (1) are converted to textual ones by the VIVO interpreter (2). The textual rules are used by the solver box (7). In (3) the user can enter a pre-constrained melody or to let VIVO create a new one according to user defined rules. The result is displayed in (4) in standard music notation. A special 'sieve' plug-in is applied to the result that connects smaller durational units–consisting of pitch repetitions–into larger ones. The basic pulse of the three lower voices can be defined in the input-score.

Finally, Figure 5 shows a sample re-harmonization of the first phrase of *Herzlich tut mich verlangen*. A quick analysis reveals several points of interest. Generally speaking, the harmonic functions appear to be correct. The movement of the voices is quite satisfactory apart from the Tenor part which is probably too agile as in this type of music the middle voices usually tend to be quite static. This could also be due to the fact that the melody is set unconventionally high. The setting is not optimal but this aspect is not covered with the rules. Likewise, the handling of the leading tone also leaves something to wish for. It has also to be noted that currently this system has no notion of a cadence. Thus, the example here was selected as it was the first run that in the end produced a cadence in a-minor. Sound examples can be found at `www.siba.fi/pwgl/demos/cmmr09-vivo.html`.

4. DISCUSSION

One of the benefits of our system is that once the VIVO database is generated it is possible to visually examine and edit it. This allows us to take away progressions we want to omit or to add new ones. It is also possible to validate the database by listening to the result. This should prove to be a quick and intuitive method of checking the integrity of the retrieval process as well. Furthermore, the system allows easily to merge databases and to add other rules when

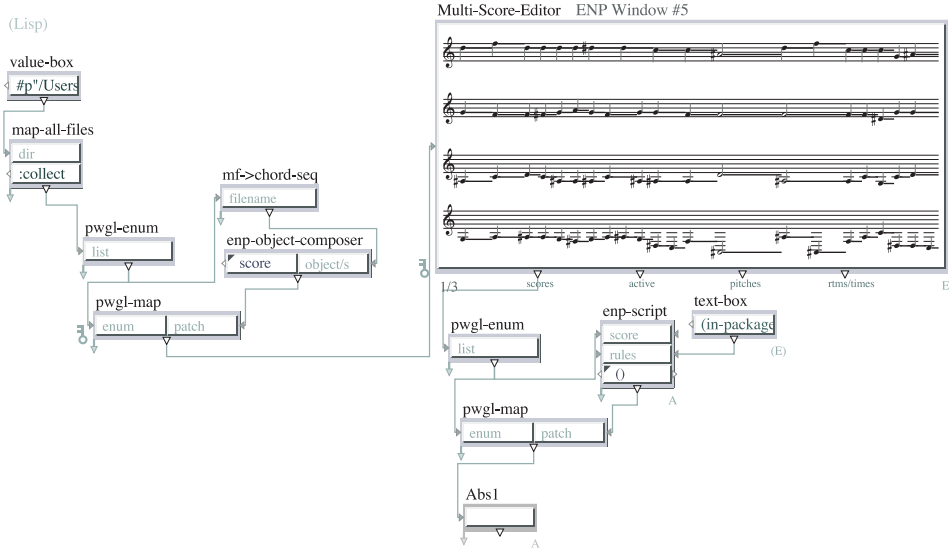

Figure 1: The basic retrieval patch.

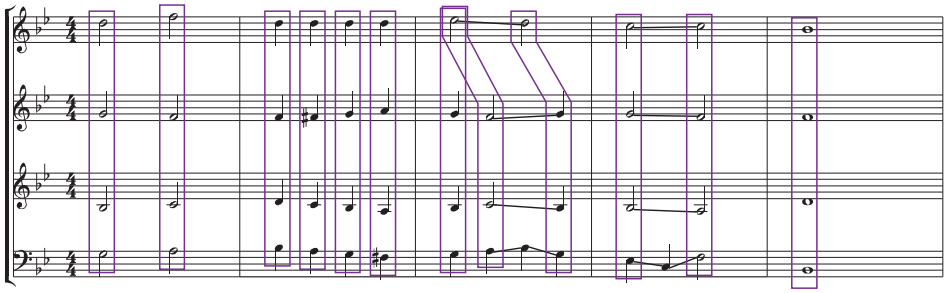

Figure 3: The rectangle shapes indicate common tertian harmonies. The line shapes indicate the presence of passing tone movement and/or incomplete tertian harmonies.

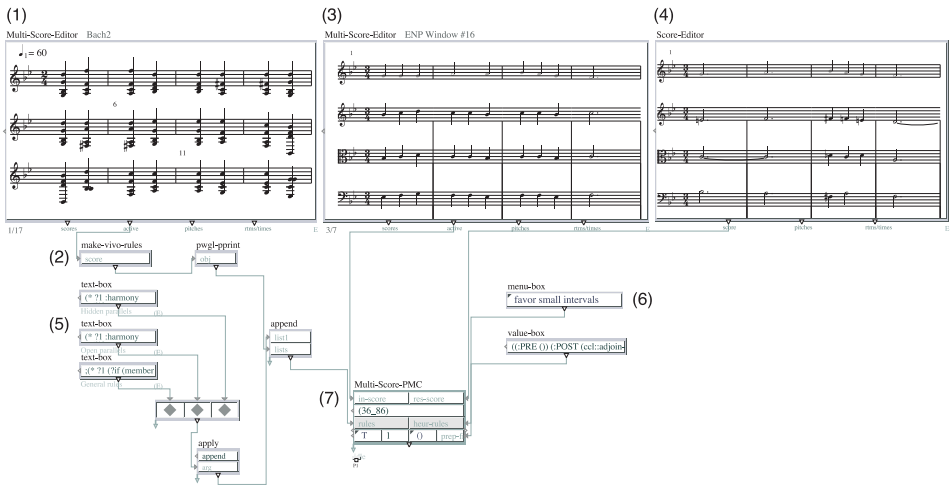

Figure 4: A VIVO patch that comprises of (1) the visual rule database, (2) the textual rule generator, (3) the input score with a pre-constrained melody, (4) the output score, (5) basic voice-leading rules, (6) heuristic rules, and (7) the solver box.

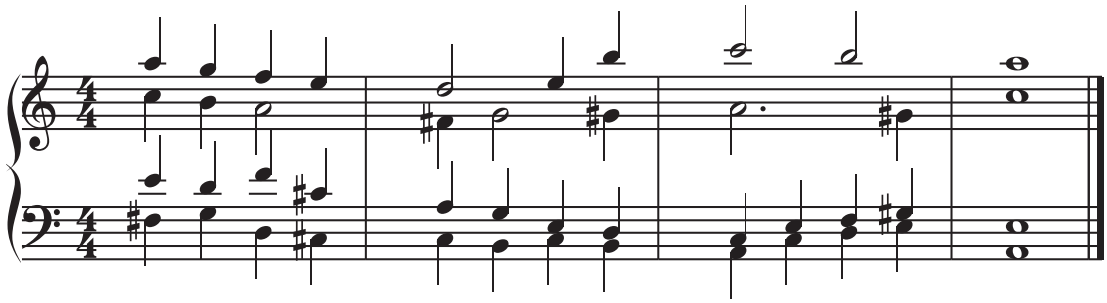

Figure 5: A sample re-harmonization of a chorale melody.

it is used as a generative tool.

The repertoire of the retrieval system should be extended to cover other aspects of the music, such as, setting, melody, etc. The retrieval process should also analyze the input material for phrases and cadences to create more convincing harmonic functions. Furthermore, our system should take advantage of other traditional MIR techniques such as pitch histograms and N-grams as considered by several authors (see for example [15]).

It is evident that in order to realize more ambitious examples a larger corpus of analyzed data should be used. The method presented here serves as a proof of concept and should be explored further. However, the re-harmonizations that the current system creates have already proven to be quite practical.

5. ACKNOWLEDGMENTS

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

6. REFERENCES

[1] Kemal Ebcioglu, "An Expert System for Harmonizing Chorales in the Style of J. S. Bach," *Journal of Logic Programming*, vol. 8, no. 1, pp. 145–185, 1990.

[2] B. Schottstaedt, "Automatic species counterpoint," Ccrma, Stanford University, 1989.

[3] Johann Joseph Fux, *Gradus ad Parnassum*, 1725.

[4] R. A. McIntyre, "Bach in a box: The evolution of four part baroque harmony using the genetic algorithm," in *IEEE Conference on Evolutionary Computation*, 1994, pp. 852–857.

[5] Johannes Feulner Hermann Hild and Wolfram Menzel, "HARMONET: A neural net for harmonizing chorales in the style of J.S. Bach," in *Advances in Neural Information Processing*. 1991, vol. 4, pp. 267–274, Morgan Kaufmann.

[6] Geraint Wiggins Dan Ponsford and Chris Mellish, "Statistical learning of harmonic movement," *Journal of New Music Research*, 1999.

[7] François Pachet and Pierre Roy, "Musical harmonization with constraints: A survey," *Constraints*, vol. 6, no. 1, pp. 7–19, 2001.

[8] S. Phon-Amnuaisuk and G. A. Wiggins, "The four-part harmonisation problem: a comparison between genetic algorithms and a rule-based system," in *AISB'99 Symposium on Musical Creativity*, 1999.

[9] Mika Kuuskankare and Mikael Laurson, "VIVO - visualizing harmonic progressions and voice-leading in PWGL," in *International Symposium on Music Information Retrieval*, 2007, pp. 289–290.

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

[11] Mikael Laurson, Mika Kuuskankare, and Vesa Norilo, "An Overview of PWGL, a Visual Programming Environment for Music," *Computer Music Journal*, vol. 33, no. 1, 2009.

[12] Mika Kuuskankare and Mikael Laurson, "Intelligent Scripting in ENP using PWConstraints," in *Proceedings of International Computer Music Conference*, Miami, USA, 2004, pp. 684–687.

[13] Mikael Laurson, *PATCHWORK: A Visual Programming Language and some Musical Applications*, Studia musica no.6, doctoral dissertation, Sibelius Academy, Helsinki, 1996.

[14] Mikael Laurson and Mika Kuuskankare, "Extensible Constraint Syntax Through Score Accessors," in *Journées d'Informatique Musicale*, Paris, France, 2005.

[15] J. S. Downie, *Evaluating a Simple Approach to Music Information Retrieval: Conceiving Melodic N-grams as Text*, Ph.D. thesis, University of Western Ontario, 1999.