From Expressive Notation to Model-Based Sound Synthesis: a Case Study of the Acoustic Guitar

Mikael Laurson[1], Jarmo Hiipakka[2], Cumhur Erkut[3], Matti Karjalainen[3], Vesa Välimäki[3], and Mika Kuuskankare[1]

[1]Sibelius Academy, Centre for Music Technology, P.O.Box 86, 00251 Helsinki, Finland

[2]Helsinki Univ. of Technology, Telecommun. Software and Multimedia Lab., Espoo, Finland

[3]Helsinki Univ. of Technology, Lab. of Acoustics and Audio Signal Processing, Espoo, Finland

E-mail: laurson@amadeus.siba.fi, Jarmo.Hiipakka@hut.fi, Cumhur.Erkut@hut.fi, Matti.Karjalainen@hut.fi, Vesa.Valimaki@hut.fi, mkuuskan@siba.fi

Abstract

The focus of this work is in modeling the unique sound and the playing practices of the acoustic guitar. The results can be applied to other plucked string instruments too. A new extended notation package is used to produce expressive control information. This tool allows the user to add to the input score instrumental expressions and tempo functions. The system includes also a rule formalism that permits further fine tuning of the computer-generated performance. A real-time synthesis engine has been developed based on earlier results in digital waveguide modeling. We also describe an analysis of vibrato in acoustic guitar tones, which provides control information for realistic synthesis.

1. Introduction

While there has been successful efforts in modeling the sound of the classical guitar [1 and 2] there is still a lack of efficient ways to play these models within a musical context. The research activity has been mostly concentrated in synthesizing isolated tones. A few attempts have been made to demonstrate different playing techniques in larger musical contexts but these trials have suffered from some fundamental problems. Typically note information has been coded tediously by hand. The sounding outcome is in this case hard to modify afterwards which causes frustrations for a musician who would want to make corrections and improvements interactively while listening to the results.

In our project the use of notation in expressive control is motivated by the lack of adequate real-time controllers, familiarity with common music notation, and precision of control. The use of common music notation requires no special technical training which in turn makes it possible to use professional players to test and verify various physical models in a deeper way than before. This kind of collaboration is of course crucial as it allows to combine the best expertise both in the technical and in the musical domains.

While common music notation provides a good starting point in expressive control it must be augmented in several ways in order to gain satisfactory results. A well known problem is for instance the fine tuning of timing during performance which has been discussed in several papers [3, 4 and 5]. Thus the user should have efficient and precise tools that allow to modify the basic rhythmical information provided by the input score. Besides standard instrument specific expressions (pizzicato, staccato, left-hand slurs) the input score must also include non-standard expressions. Non-standard expressions are usually not shown in ordinary notation but it is anyway assumed that a good player adds them according to her/his taste and the character of the piece.

2. Expressive Notation Package (ENP)

We have developed a new notation package (Expressive Notation Package, ENP) to control the model-based synthesis engine. ENP is a PatchWork (PW, [6 and 7]) user library and it is written in Lisp and CLOS (Common Lisp Object System). ENP's object-oriented approach makes the system extendible and open. As ENP is built on top of PW, a rich set of tools is available when preparing scores. These allow the user to produce pitch, rhythm and textural information either algorithmically or by hand. ENP resembles commercial notation packages since it requires no textual input. Besides a full set of standard notation facilities, ENP has user-definable extensions that allow efficient description of interpretation.

2.1 Expressions

Expressions can be applied to a single note (such as string number, pluck position, vibrato, or dynamics) or to a group of notes (left-hand slurs, finger-pedals). Groups can overlap and they may

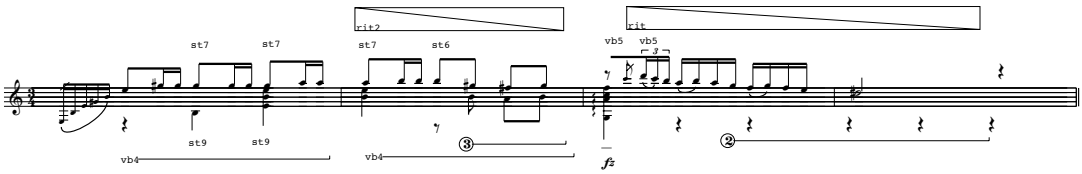

Figure 1. Musical excerpt with ENP expressions.

contain other objects, such as breakpoint functions. The latter case is called a group-BPF. Macro expressions generate additional note events (tremolo, trills, portamento, rasgueado).

Fig. 1 gives an ENP example— measures 25-28 from "Madroños" for guitar by Federico Moreno Torroba—which includes both standard instrumental expressions and non-standard ones.

2.2 Fine-tuning of Timing

ENP allows fine-tuning of timing with the help of graphical tempo functions. In order to assure synchronization of polyphonic scores, all tempo functions are merged and translated internally into a global time-map [3]. Tempo modifications are defined by first selecting a range in the score where the tempo change should occur. After this a group-BPF is applied to the range. The group-BPF can be opened and edited with the mouse. In order to facilitate the edition process, the group-BPF editor displays — besides the tempo function — the selected music notation excerpt in grey-scale in the background (see Fig. 2).

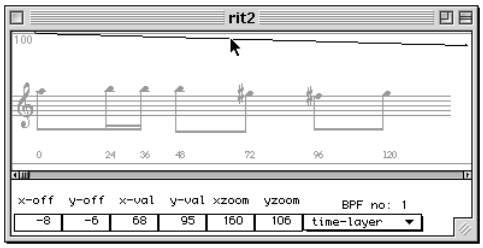

Figure 2. Edition of a tempo function.

Tempo function values are given in percentage (i.e. 100 means 'a tempo', 200 twice as fast, 50 half tempo). For instance an accelerando from 'a tempo' to 'twice as fast' is achieved with a ramp that starts from 100 and that ends at 200.

In addition to conventional accelerandi and ritardandi, the user can apply special rubato effects ("give and take") to a group. This mode is especially useful when the resulting tempo starts to fluctuate too wildly. As in the previous case the user starts with a selection in the score and applies a tempo function to it. The difference is though that the duration of the range is not affected by the time modification. Time modifications are only effective inside the selected range.

2.3 Performance Rules

Besides tempo functions, ENP supports user definable performance rules which allow to modify score information in a similar way as in the Swedish "Rulle" system [4 and 5]. In Rulle, performance rules are used to calculate timing information, dynamics and other synthesis parameters. The main difference is though that ENP rules use a syntax which was originally designed for PWConstraints (for more details see [7 and 8]).

ENP rules are written in two parts: (1) a pattern-matching part which is followed by (2) a Lisp code part. The pattern-matching part checks when the rule should be applied and also extracts relevant note information which is used later by the Lisp code part. The Lisp expression, in turn, executes the actual alteration.

For instance a variant of the well known "notes inégales" rule ("in a beat of two eighth-notes the first note is made longer than the second one") can be translated into PWConstraints syntax as follows:

```
;; 1. pattern-matching part
(* ?1 ?2  (rtm-match? (1 ((?1 1) (?2 1)))
;; 2. Lisp-code part
(?if (write-key ?1 :dr
        (+ (read-key ?1 :dr) 0.1))
     (write-key ?2 :dr
         (- (read-key ?2 :dr) 0.1)))))
```

The pattern-matching part (line `1.`) states that if two notes (`?1` and `?2`) are adjacent and form a two eighth-note beat, then the first note (`?1`) is lengthened by 0.1 s and the second one (`?2`) shortened by 0.1 s.

2.4 Calculation of Control Data

The calculation of the control information for the model-based synthesis engine is executed in two main steps. In the first one, the note information provided by the input score is modified by the tempo functions and ENP performance rules. Furthermore some instrument specific rules (in our case the classical guitar) are applied which further modify the input score.

In the second step, all notes of the input score are scheduled. While the scheduler is running, each note sends a special method to its instrument which in turn starts other scheduled methods which typically produce the final control data. These methods are responsible for creating discrete control data (such as excitation information) or

continuous data (gain of the loop filter, filter coefficients, or other low-level data).

If a note contains a macro expression (trill, portamento, rasgueado) then new note instances are created on the fly by instrument specific methods and each new note is inserted in the scheduler queue.

Currently ENP is able to produce control information either as MIDI data or as a text file.

3. Guitar Synthesis Model

We have implemented a real-time, model-based guitar synthesizer using workstation computers. The plucked-string synthesis model is based on the theory of digital waveguide modeling, primarily developed by Smith [9].

The guitar model incorporates six dual-polarization string models, so that effects such as two-stage decay and beating [1, 10], resulting from the differences in the string's vibratory motion parallel and perpendicular to the soundboard can be accounted for in the synthesis.

Each vibration polarization is modeled as a simple single-delay-loop (SDL) string model (for a derivation of SDL string models, see [9 and 10]). In the SDL model, a delay line and a fractional delay filter together determine the fundamental frequency of the synthesized tone [1]. A one-pole lowpass filter, called the loop filter, determines the characteristics of the frequency-dependent attenuation of the synthetic tone [1]. The input signal to the string models is a wavetable signal filtered according to plucking position and style [10].

The string model includes also input and output for sympathetic coupling implementation between strings. The sympathetic coupling output signals are fed through a coefficient matrix which determines the strength of the coupling [10]. This structure is inherently stable, regardless of the values of the coupling coefficients.

4. Extraction of Control Parameters

Since our aim is realistic and expressive computer-generated performance, we have recorded acoustic guitar sounds in an anechoic chamber, and analyzed various control parameters from the sound signals. The loop-filter parameters for each string and fret position were estimated using the method described by Välimäki *et al*. [1]. The excitation signals were extracted according to the technique proposed by Välimäki and Tolonen [11].

Other analyzed features are related to different plucking styles (apoyando, tirando, and rasgueado), vibrato, re-plucking of a string that is already sounding, damping of tones, and portamento effects. Furthermore, dynamic variations of different control parameters were measured to obtain information of how much tone-to-tone variations there are in musical pieces performed by a professional guitar player. The coupling parameters for the model of sympathetic vibrations were also estimated from the recordings.

As an example, we document the analysis results of vibrato in acoustic guitar tones. The player was asked to perform slow and fast vibrato on various strings and fret positions. The time-varying fundamental frequency of each tone was then analyzed using the autocorrelation method. Fig. 3 shows examples of typical fast and slow vibrato. Note that in both cases the fundamental-frequency variation converges to a nearly sinusoidal waveform soon after the starting time t_0. We call the time between the starting time and the converge transient time t_t. Its value is typically around 0.5 s. According to our analysis the lowest frequency obtained during vibrato is the nominal fundamental frequency of the tone without vibrato. The maximum deviation of the fundamental frequency depends on the string and fret. The starting time of vibrato naturally depends on the score. For the professional player used in our experiments, the mean vibrato rates were 1.4 Hz and 4.9 Hz for the slow and the fast vibrato, respectively. The amplitude of vibrato is systematically smaller in the fast case than in the slow one.

In the sound synthesis, we can generate vibrato by deviating the delay-line length with a sine wave plus a small amount of lowpass filtered noise—to bring about the small random fluctuations seen in Fig. 3. The frequency and amplitude of the sine wave may be set according to our analysis results.

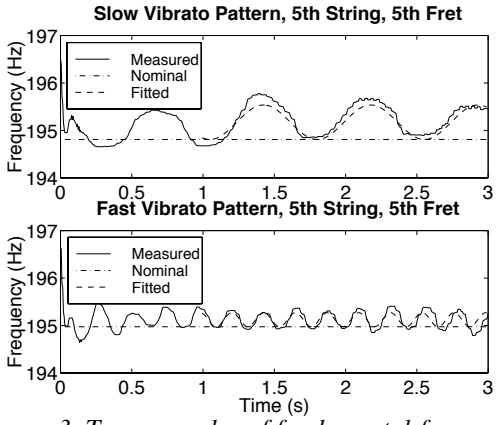

Figure 3. Two examples of fundamental-frequency deviations during vibrato.

5. Control Protocol

We have designed a control protocol that allows detailed control of the advanced guitar model in a portable way. Our protocol has no MIDI-like limitations in addressing and parametrization.

5.1 Addressing Scheme

The addressing scheme of our protocol is a hierarchical system, quite similar to the familiar URL-addresses and file-system structures. Any address path in our system consists of a slash-

separated list of identifiers, with the network address as a prefix.

The addressing scheme makes it feasible to divide the synthesis model or an ensemble of models into functional substructures that are controlled independently. In the case of the guitar model, natural substructures are the string models and the guitar body model. A reference to the first string of the guitar model may begin, e.g., with the string 'guitar/string1/', followed by a parameter or an operation name. Similarly 'guitar/body/ res1/freq' may be the path of the center frequency of the first separate body model resonator.

When the control protocol is used over a network interface, the network address of the sound synthesizing host may be appended in front of the parameter address, using the format '//net.address.here:port/'. The protocol assumes that the network transport layer provides a back-channel so that acknowledgement messages and later perhaps the synthesized sound itself can be sent back to the controller device. A rather similar addressing mechanism has been previously presented by Wright and Freed [12], but without the network address part.

5.2 Parameters

The hierarchical control protocol makes it possible to control an arbitrary number of different parameters. Most of the controllable parameters in our guitar model are directly related to the DSP-level parameters of the model. The parameter names are intended to be clear enough to be human-readable. Also the parameter values are transferred as ASCII strings and conversion is thus necessary in the receiving end.

The parameter names can generally be arbitrary. Some parameters, however, have a special meaning in our system, in that they interact across the hierarchy levels. An example of such parameters is 'amplitude', which must be implemented by the synthesis system so that the 'amplitude' values cumulate. The same approach has previously been described for the ZIPI proposal [13].

Acknowledgements

This collaboration between Sibelius Academy and Helsinki University of Technology has been made possible by the funding provided by the Academy of Finland. The authors are grateful to Mr. Klaus Helminen who played the acoustic guitar samples used in this study.

References

[1] V. Välimäki, J. Huopaniemi, M. Karjalainen, and Z. Jánosy. Physical Modeling of Plucked String Instruments with Application to Real-Time Sound Synthesis. *J. Audio Eng. Soc.*, Vol. 44, No. 5, pp. 331-353, May 1996.

[2] Z. Jánosy, M. Karjalainen, and V. Välimäki. Intelligent Synthesis Control with Applications to a Physical Model of the Acoustic Guitar. In *Proc. ICMC'94*, pp. 402-406, Aarhus, Denmark, Sept. 1994.

[3] D. Jaffe. Ensemble Timing in Computer Music. *Computer Music J.* , Vol. 9, No. 4, pp. 38-48, 1985.

[4] A. Friberg. Generative Rules for Music Performance: A Formal Description of a Rule System. *Computer Music J.*, Vol. 15, No. 2, pp. 49-55, Summer 1991.

[5] A. Friberg, L. Frydén, L.-G. Bodin and J Sundberg. Performance Rules for Computer-Controlled Contemporary Keyboard Music. *Computer Music J.*, Vol. 15, No. 2, pp. 56-71, 1991.

[6] M. Laurson and J. Duthen. PatchWork, a Graphical Language in PreForm. In *Proc. ICMC'89*, pp. 172-175, 1989.

[7] M. Laurson. PATCHWORK: A Visual Programming Language and Some Musical Applications. Doctoral dissertation, Sibelius Academy, Helsinki, Finland, 1996.

[8] M. Laurson. PWConstraints. Reference Manual. IRCAM, Paris, France, 1996.

[9] J. O. Smith. Physical Modeling Using Digital Waveguides. *Computer Music J.*, Vol. 16, No. 4, pp. 74–91, 1992.

[10] M. Karjalainen, V. Välimäki, and T. Tolonen. Plucked-String Models: From the Karplus-Strong Algorithm to Digital Waveguides and Beyond. *Computer Music J.*, Vol. 22, No. 3, pp. 17-32, Fall 1998.

[11] V. Välimäki and T. Tolonen. Development and Calibration of a Guitar Synthesizer. *J. Audio Eng. Soc.*, Vol. 46, No. 9, pp. 766-778, Sept. 1998.

[12] M. Wright and A. Freed. Open SoundControl: A New Protocol for Communicating with Sound Synthesizers. In *Proc. ICMC'97.,* pp. 101-104, Thessaloniki, Greece, Sept. 1997.

[13] K. McMillen, D. L. Wessel, and M. Wright. The ZIPI Music Parameter Description Language. *Computer Music J.*, Vol. 18, No. 4, pp. 52-73, 1994.