The Bowed Tube: a Virtual Violin

Alfonso Perez Carrillo
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
alfonso.perez@upf.edu

Jordi Bonada
Music Technology Group
Universitat Pompeu Fabra
Barcelona, Spain
jordi.bonada@upf.edu

ABSTRACT

This paper presents a virtual violin for real-time performances consisting of two modules: a violin spectral model and a control interface. The interface is composed by a sensing bow and a tube with drawn strings in substitution of a real violin. The spectral model is driven by the bowing controls captured with the control interface and it is able to predict spectral envelopes of the sound corresponding to those controls. The envelopes are filled with harmonic and noisy content and given to an additive synthesizer in order to produce violin sounds. The sensing system is based on two motion trackers with 6 degrees of freedom. One tracker is attached to the bow and the other to the tube. Bowing controls are computed after a calibration process where the position of virtual strings and the hair-ribbon of the bow is obtained. A real time implementation was developed as a MAX/MSP patch with external objects for each of the modules.

Keywords

violin, synthesis, control, spectral, virtual

1. INTRODUCTION

In this paper we present a real-time controller, the *Bowed-Tube*, that captures bowing gestures and synthesizes violin sounds. The system is composed of two main modules: a control interface that captures gestural data during the performances and a spectral model driven by gestures.

There is a big interest in violin-like control interfaces due to a great extent on the high gestural degree of bowing. Some of those systems are dedicated to evaluate violin synthesis models [8, 12] and most of them propose extensions for the violin or make use of them as pure controllers for any kind of event [5, 11, 1]. In this work we use the interface to imitate a violin sound, but as the controller and synthesis modules are independent, the control parameters could be mapped to any other sound.

Existing real time synthesizers driven by gestures are almost restricted to physical models [2, 9]. In this work we present a spectral model able to predict spectral envelopes given a set of bowing gestures. A similar approach is used in [7].

NIME2010, June 15-18, Sydney, Australia

The general behavior of the system is depicted in Figure 1: Gestural data is captured with the control interface and a set of descriptors (e.g. bow velocity, bow force) is computed from it. These descriptors are passed to the spectral model that generates spectral envelopes, which are in turn filled with harmonic and noisy content and processed b y the additive synthesizer in order to obtain a synthetic string vibration signal. At the end, the output of the additive synthesizer is convolved with a violin body impulse response, obtaining the final sound.

The control interface consists of a sensing violin bow and a tube with drawn strings (Figure 2). The sensing system is based on [4], which was developed to capture bowing gestures from real violin performances. The system has two motion trackers, one attached to the bow and one to the tube, which allow to keep track of the position of the the hair ribbon and the virtual strings respectively. Knowing the position of bow and strings we can compute a set of bowing descriptors that is used as input for the spectral model.

The spectral model is a generative model based on neural networks and was trained with a multimodal database of violin performances with gestural and audio information. The gestural information was measured with a similar sensing system adapted to a real violin and audio is captured as string vibration. The model is able to predict spectral envelopes of the sound corresponding to certain bowing parameters [6]. These envelopes are used later as input of an additive synthesizer that produces the final violin sound.

Real-time implementation to show the system working was developed in the MAX/MSP [1] framework.

2. CONTROL INTERFACE

The control interface consists of a sensing violin bow and a tube with four drawn strings (see Figure 2). Bow and tube motion is captured with a commercial 3D motion tracking system (*Polhemus Liberty system* [2]). The system consists of a *source*, an approximately 10x10x10 cm cube which generates an electro-magnetic field (EMF), a set of small sensors or *trackers* and a signal processing box to which source and sensors are wire-connected. When the source is emitting the EMF, the system can determine the position and orientation of each sensor inside the magnetic field. The sample rate is 240 Hz, workable-range around 1.5 meters and accuracy depends on the distance to the source and the velocity of the movement, being the static position accuracy of around 0.71 mm and static orientation accuracy of 0.15 degrees.

[1] MAX/MSP Tools for Media
`http://cycling74.com/`
[2] Polhemus Motion Tracking Sensors.
`http:\\www.polhemus.com`

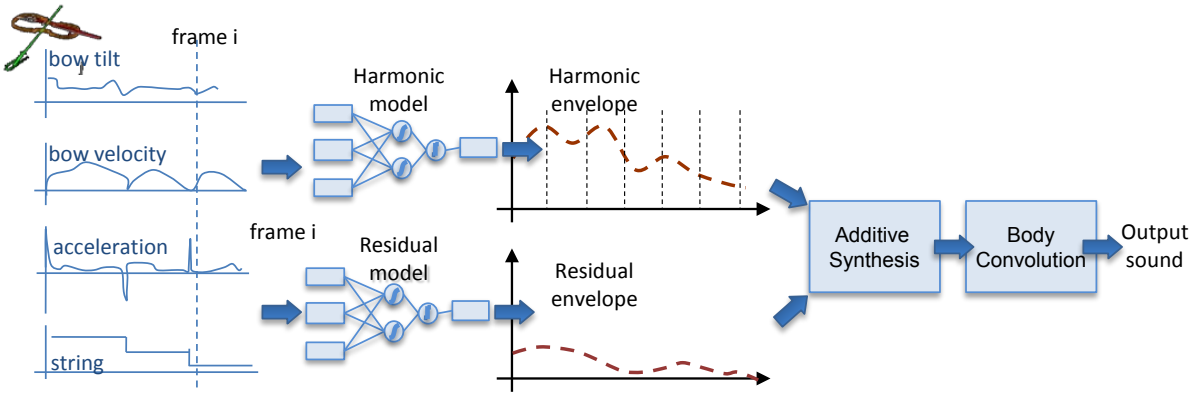

Figure 1: General Scheme. The gestures captured with the sensing system drive a spectral model that produces the corresponding spectral envelopes that are filled with harmonic and residual components and used for additive synthesis. At the end a body impulse response is convolved obtaining the final violin sound.

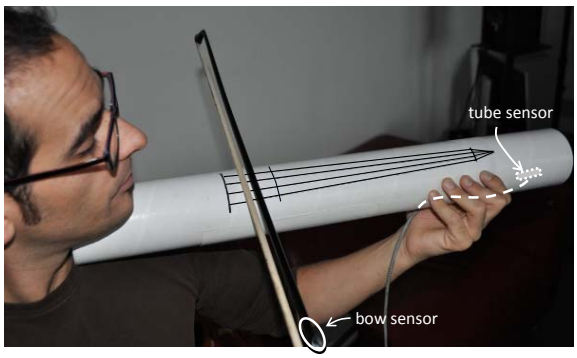

Figure 2: Playing the virtual violin. One sensor is attached to the bow and other to the tube.

We make use of three sensors, the first one is attached to the back of the tube. The second one is attached to the bow-stick trying to keep the balance position of the bow (see Figure 2). An additional *stylus* sensor with a needle metal tip is used as a pointer for the calibration. The system and calibration is based following a similar procedure as in [4].

2.1 Sensor Calibration

Calibration is needed to know the position of the virtual strings with respect to the tube sensor and the position of the hair ribbon with respect to the bow sensor. The process consists of 'sampling' the string ends and hair ribbon ends with the *stylus* sensor. Relative coordinates of these points are constant during the whole performance as the tube and the bow are (almost) rigid objects. For any position and rotation of the sensors at any time of the performance, we can obtain the coordinates of the calibrated points with respect to the *source* coordinate system, by applying basic geometric operations. This way we know the position of the strings and bow hair ribbon with respect to the same coordinate system.

A second step in the calibration consists of finding the corresponding inclination of the bow when playing each of the strings. This is done by computing the angle between the line representing the hair-ribbon and the plane defined with three of the string ends. Three boundary angles are annotated, one for each transition between two consecutive strings (i.e. G-D, D-A, A-E). A hysteresis of one degree is applied in order to avoid glitches at the string boundaries. Angle calibration determines the string being played and this parameter determines the point of contact between bow and string (p_C). All computations of the bowing parameters are based on this parameter and on the string being played.

2.2 Bowing Descriptors

The following bowing descriptors are necessary inputs for the spectral model (Section 3.1).

- **Bow transversal position** (*bpos*), or just bow position. It is computed as the Euclidean distance between the beginning of the hair ribbon and p_C. When computing this descriptor, we always consider the A string as the playing string in order to avoid discontinuities caused by string changes.
- **Bow transversal velocity** (*bvel*). It is computed as the derivative of the bow position $\frac{d}{dt}bpos$.
- **Bow transversal acceleration** (*bacc*). It is computed as the derivative of velocity $\frac{d}{dt}bvel$.
- **Bow to bridge distance** (*bbd*), is computed as the Euclidean distance between p_C and the beginning of the string.
- **Bow pressing force estimate** ($bforce_{est}$). As calibrated strings and hair ribbon are rigid, when pressing the bow on the tube, the computed hair ribbon line becomes below the string line. The shortest distance between string and hair lines is proportional to the bow force applied.
- **Playing state** (*isPlaying*). If both $bforce_{est}$ and *bvel* are positive, then the state is *playing* otherwise it is *not-playing*.
- **Bow tilt**. It is the angle around the bow-stick axis.

An estimation of the **pitch** is also necessary as input for the spectral model. It can be obtained in several ways. The most simple way is by using a real violin as controller instead of the tube (i.e. analyzing the pitch of an electric violin signal). However, we are using the tube and therefore we need another pitch-controller. One solution is to sing and use a microphone and another possibility is to map any gestural descriptor (i.e. bow position) to pitch. If no pitch is computed, we can assign a fix pitch to each string but ornaments such as vibrato will not be possible.

2.3 Additional Descriptors

Many other descriptors can be computed, not necessary for imitating a real violin but as an extension to it. Just as an example, we list below of some them being used for a musical piece in work that makes use the same interface.

- **Bow skewness**. It is the angle between bow-stick and the bridge.
- **Bow-string distance**. It is actually the same as the force estimator, but when the bow is not in contact with the string.
- **Bow acceleration**. Acceleration in the direction of the movement. This descriptor can be used in order to trigger events.
- **Bow-sensor 3D position**. Sounds can be controlled depending on the position of the sensor attached to the bow.
- **Violin azimuth and elevation**. Considering the violin as a vector starting at the bridge and going along one of the strings, the violinist can point with the violin to any direction and could be used to control any effect, for instance panning.

3. SYNTHESIS

The synthesis is driven by the bowing gestures. It is a procedure with three main parts: 1) generation of spectral envelopes given the bowing parameters; 2) fill the envelopes with harmonic and residual content and apply additive synthesis and 3) convolution of the additive synthesis with a violin body impulse response. This convolution is necessary because the neural networks are actually modeling the string vibration instead of the acoustic sound of the violin. More details can be found in [6].

3.1 Violin Spectral Model

A violin timbre model based on neural networks is able to predict spectral envelopes corresponding to the sound that would be produced by the set of given bowing controls. To build the model, it is trained with a database of gestures and audio descriptors captured during real violin performances with a similar sensing system as the described in Section 2. The model is able to predict spectral envelopes for the harmonic and residual components of the sound. Both envelopes are given to an additive synthesizer (Section 3) which produces the violin sound. More details about the timbre model can be found in [6].

3.2 Additive Synthesizer

Predicted envelopes are used for synthesizing violin sounds following an additive synthesis approach based on SMS [10]. For each frame, harmonic envelopes are filled with harmonic content (determined by the pitch) and residual envelopes are filled with noise and both summed up obtaining the corresponding spectrum. Spectra are transformed to the temporal domain by applying an IFFT, and these temporal frames are overlap-added in order to obtain the synthetic sound. In a traditional violin, during note attacks the non-linear friction produces a variety of chaotic waveforms which are perceptually important. We are not able to predict these transients but the predicted noise envelope allows for the retention of the temporal energy distribution of the transients.

3.3 Body Impulse Response

The signal modeled with the timbre model is the signal captured with a bridge pickup, very similar to string velocity signal. In order to obtain a realistic violin sound from this kind of signal, it has to be convolved with a body impulse response obtained as reported in [6]. In the same manner, the output of the additive synthesis has to be convolved with the same impulse response.

4. MAX/MSP IMPLEMENTATION

The presented interface is implemented in the framework MAX/MSP. Three external objects have been developed in C: *polhemusTracker*, *simpleNN* and *additiveSynth* which are explained below.

A MAX/MSP patch is shown in Figure 3. The left side of the figure shows the data acquisition part (bowing gestures and pitch), and the right side shows how input descriptors are given to the Neural Networks (*simpleNN* objects) that generate the spectral envelopes. These envelopes are passed to the additive synthesizer (*additiveSynth*$\sim$ object), which produces the synthetic sound that is sent to the signal out line (*dac*$\sim$ object).

A video of a performance with the virtual violin can be found online [3]. The first part of the video shows the system working on a real violin in order to be able to detect the pitch, which is a very important parameter in violin playing specially when using vibrato. We can listen the synthetic sound (is not coming from the violin) with and without body convolution. In the final part of the video we show the bowed violin and we can listen to a short performance. Note that pitch is fixed for each string and there is no vibrato.

4.1 *polhemusTracker*

The *polhemusTracker* (Figure 3) object is responsible for connecting to the Polhemus sensors (message *start* connects with the tracker and message *stop* disconnects), for capturing the raw motion data and for computing the bowing descriptors. The frame acquisition rate can be specified with the message *sampleRate* having its maximum value determined by the Polhemus system sample rate (240Hz).

Computed descriptors are visualized in real time with *multisliders* objects. Pitch is detected analyzing the line-in signal (*adc*$\sim$) with the help of the *fiddle*$\sim$ object [4] . If pitch is not being detected (e.g. there is no microphone) fixed frequencies are assigned to each string (in our case we assigned the frequencies corresponding to a standard violin tuning E=670Hz, A=444Hz, D=295Hz and G=198Hz).

4.2 *simpleNN*

The *simpleNN* object (Figure 3) implements the neural networks of the timbre model. The object takes as parameter in the constructor the name of the model (i.e. *timbre_rmsRel41*). The object looks for a configuration file with that model name. In that file are specified the input parameters and the structure of the neural network. The number and type of the inlets are are created depending on the specifications. The output is the predicted spectral envelope, which is visualized in a *multislider*. A spectral envelope is represented as the energy in 40 frequency bands that follow a logarithmic scale and therefore the *multislider* has 40 points. Two *simpleNN* objects are used, one for predicting the harmonic envelopes and the other for the residual envelopes.

[3] `http://www.youtube.com/MusicTechnologyGroup#p/u/0/NM50dEXRQVk`

[4] fiddle$\sim$ object by Miller Puckette `http://crca.ucsd.edu/~tapel/software.html`

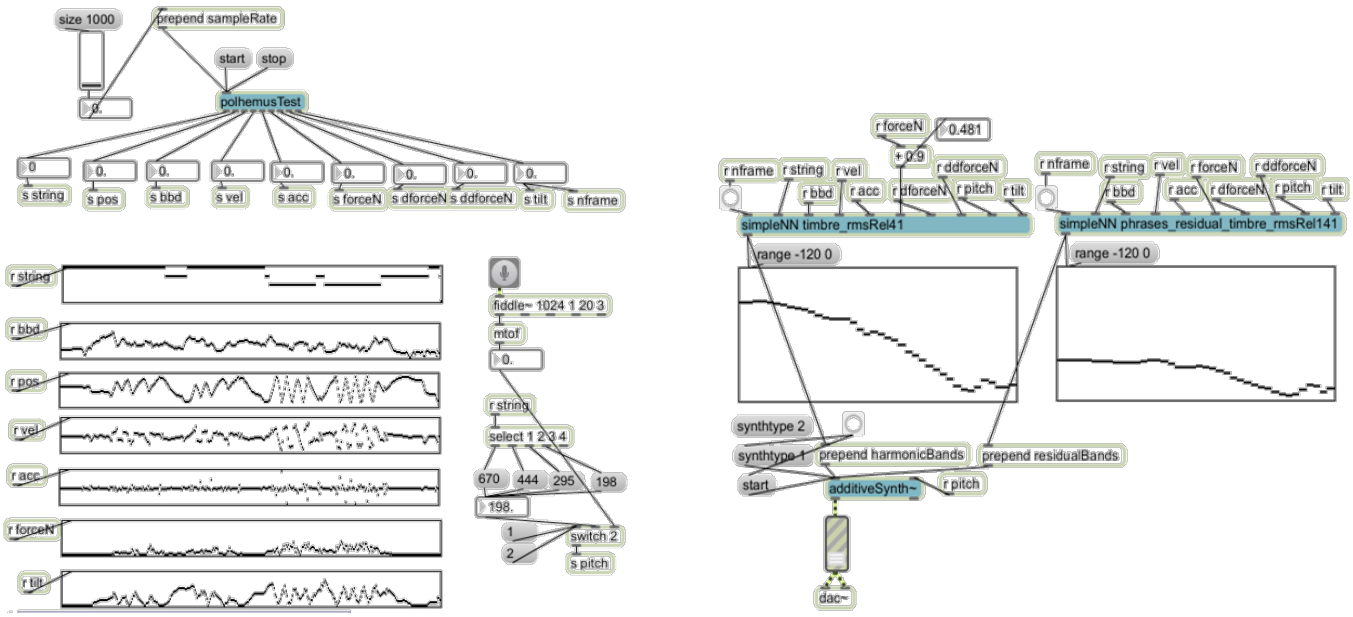

Figure 3: MAX/MSP patch with the three externals: the sensor interface *polhemusTracker* (in the figure appears as polhemusTest), the Neural Networks (*simpleNN*) and the synthesis module (*additiveSynth~*)

4.3 *additiveSynth~*

The predicted energy at the 40 the frequency bands of the two *simpleNN* objects is given to the *additiveSynth* object (Figure 3) that is responsible for filling the envelopes with harmonic and noisy content, summing them up, performing the IFFT and doing the overlap-add operation in time. The harmonic content of the envelopes is determined by the current pitch which is an input as well. The additive synthesizer is an adaptation of *smspd* [3], a Pure Data implementation of SMS [10].

5. CONCLUSIONS

We presented a real-time virtual violin controlled with a sensing bow and a tube with 4 drawn strings. The system imitates the sound of a real violin based on gestural data. The intention in this paper was to simulate the sound of a violin with a similar interface, but there are many possibilities to build an extended violin interface, for instance, by using the described additional descriptors or by simply adding extra strings. Furthermore, the controller does not need to be used with a violin synthesizer but used as a controller for any application. The spectral model introduced represents one of the main differences with respect to other virtual violins which are almost restricted to physical models. Additionally we provide an online video to show the system working.

6. ACKNOWLEDGMENTS

The authors would like to thank Esteban Maestre, Merlijn Blaauw and Enric Guauss for their help building the sensing device.

7. REFERENCES

[1] F. Bevilacqua, N. Rasamimanana, E. Fléty, S. Lemouton, and F. Baschet. The augmented violin project: research, composition and performance report. In *Proceedings of the International Conference on New Interfaces for Musical Expression (NIME)*. 2006.

[2] M. Demoucron. *On the control of virtual violins: Physical modelling and control of bowed string instruments*. PhD thesis, Universite Pierre et Marie Curie (Paris, France) and the Stockholm Royal Institute of Technology (Stockholm, Sweden), 2008.

[3] R. T. Eakin and X. Serra. Smspd, libsms and a real-time sms instruments. *International Conference on Digital Audio Effects*, 1/9/2009 2009.

[4] E. Maestre, J. Bonada, M. Blaauw, A. Pérez, and E. Guaus. Acquisition of violin instrumental gestures using a commercial EMF device. In *Proc.Int. Computer Music Conf.*, Copenhagen, Denmark, 2007.

[5] C. Nichols. The vbow: development of a virtual violin bow haptic human-computer interface. In *NIME '02: Proceedings of the 2002 conference on New interfaces for musical expression*, pages 1–4, Singapore, Singapore, 2002. National University of Singapore.

[6] A. Perez. *Enhancing Spectral Synthesis Techniques with Performance Gestures using the Violin as a Case Study*. PhD thesis, Universitat Pompeu Fabra, 2009.

[7] B. Schoner. *Probabilistic Characterization ans Synthesis of Complex Driven Systems*. PhD thesis, MIT Media Lab, 2000.

[8] E. Schoonderwaldt and M. Demoucron. Extraction of bowing parameters from violin performance combining motion capture and sensors. *J. Acoust. Soc. Am.*, in press.

[9] S. Serafin. *The sound of friction:real-time models, playability and musical applications*. PhD thesis, CCRMA, Stanford University, CA, USA., June 2004.

[10] X. Serra. *A System for Sound Analysis/Transformation/Synthesis based on a Deterministic plus Stochastic Decomposition*. PhD thesis, Stanford University, 1989.

[11] D. Trueman. Reinventing the violin.

[12] D. S. Young. Wireless sensor system for meassurement of violin bowing parameters. In *Proceedings of the SMAC*, 2003.