Physical Modelling and Supervised Training of a Virtual String Quartet

Graham Percival
School of Engineering
University of Glasgow, UK
graham@percival-music.ca

Nicholas Bailey
School of Engineering
University of Glasgow, UK
nick@n-ism.org

George Tzanetakis
Dept. of Computer Science
University of Victoria, Canada
gtzan@cs.uvic.ca

ABSTRACT

This work improves the realism of synthesis and performance of string quartet music by generating audio through physical modelling of the violins, viola, and cello. To perform music with the physical models, virtual musicians interpret the musical score and generate actions which control the physical models. The resulting audio and haptic signals are examined with support vector machines, which adjust the bowing parameters in order to establish and maintain a desirable timbre. This intelligent feedback control is trained with human input, but after the initial training is completed, the virtual musicians perform autonomously. The system can synthesize and control different instruments of the same type (e.g., multiple distinct violins) and has been tested on two distinct string quartets (total of 8 violins, 2 violas, 2 cellos). In addition to audio, the system creates a video animation of the instruments performing the sheet music.

Categories and Subject Descriptors

H.5.5 [**Information Interfaces and Presentation**]: Sound and Music Computing; I.2.1 [**Artificial Intelligence**]: Application and Expert Systems

General Terms

Algorithms, Design, Human Factors

Keywords

Violin synthesis, physical modelling, intelligent feedback control, machine learning, autonomous music performance

1. INTRODUCTION

Computer synthesis of music performed by bowed string instruments is a challenging problem. Unlike instruments whose notes originate with a single discrete excitation (e.g., piano, guitar, drum), bowed string instruments are controlled with a continuous stream of excitations (i.e. the bow scraping against the string). Many existing synthesis methods utilize recorded audio samples. Although sampling performs reasonably well for single-excitation instruments, the results are not as good for continuous-excitation instruments.

There are a number of applications of computer synthesis of music. The main three are: interactive performance with novel interfaces, preparing an audio track with manual adjustments to the synthesis, and autonomous performance of music with no human input. It is the latter goal that this work focuses on; a typical example would be to allow a composer to quickly hear a rough performance of her work in progress, illustrated in Figure 1. Other applications could be to create audio and video as part of a digital violin tutor [37], or to allow users of a digital music library [10] to hear what the sheet music sounds like without requiring an audio recording; this could greatly mitigate copyright concerns.

Autonomous performance of music is tested in the Musical Performance Rendering Contest for Computer Systems [16], in which researchers write programs to perform piano music (generally Mozart or Chopin) which are then performed on a disklavier (a computer-controlled acoustic piano). In contrast, our work focuses on string quartets instead of piano.

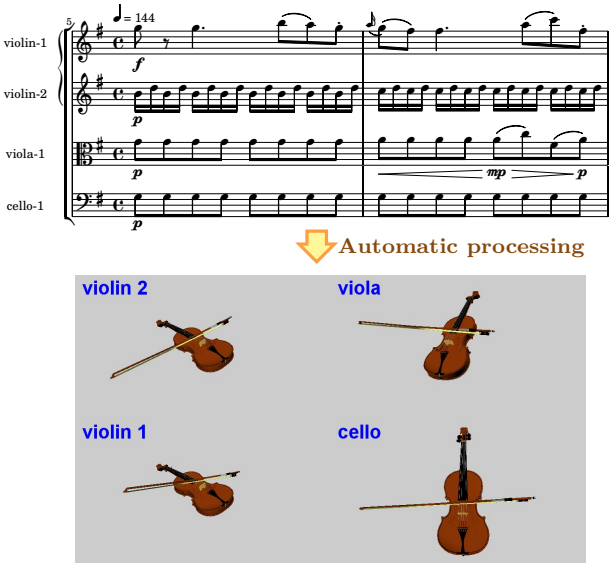

Figure 1: Video produced automatically from the sheet music, performed by two distinct string quartets: `http://percival-music.ca/mm2013.html`

ACM Multimedia 2013, Barcelona, Spain

1.1 Related Work

One method of synthesizing music in music notation programs is sample-based synthesis: a selection of samples from a library of recorded audio are concatenated. This process can be enhanced in various ways, such as Reconstructive Phrase Modeling [18] or Spectral Modelling Synthesis (SMS) [31], in which instead of concatenating the time-domain data directly, the samples are first analyzed in some manner and then reconstructed. For example, in SMS, a spectral analysis of the samples results in a set of partials plus a residual signal which is treated as filtered noise. A re-synthesis of these partials plus noise captures the transients at the beginning of notes much better than a technique relying on partial-based synthesis alone. SMS has been used to great effect in the Vocaloid singing synthesis program [17].

Analysis and resynthesis of recorded audio can also be used to create a virtual instrument which is then controlled with a virtual performer which creates a stream of control data, such as trumpet performances [7]. A similar approach was used to create violin performances: a neural network was used to predict spectral data based on (potentially) simulated physical actions [24]; the resulting virtual violin was then performed by recording physical actions of real violinists and re-synthesizing those recorded actions [19].

Another method of generating audio is by modelling the physics of the instrument in response to various actions (e.g., bow force for violin, air pressure for clarinet, strike position for drums). The mechanics of musical instruments is a well studied topic in acoustics and physical modeling synthesis uses computational techniques to simulate the physics of sound generation in instruments. A good overview of physical modeling is presented in [33] while the standard library is the Synthesis ToolKit in C++ [26, 32]. There are two main problems with physical models: they are more computationally intensive than most sample-based synthesis methods, and they are difficult to control. One early method of performing music with a physical model used a hierarchical set of expert systems which played a simulated trumpet [6]. Violin music has been created by recording physical actions from real violinists and re-synthesizing the actions with a physical model [9].

Assuming that the physical model acts similar to a real instrument, important clues to its performance can be gained by studying the actions of skilled musicians. Various systems have been created to record actions of real violinists (notably bow force, velocity and distance from bridge, although some projects recorded other factors) [9, 19, 38, **?**], for either the analysis of musical performances or resynthesis of audio. Performing music by concatenating segments of these recordings is limited by three factors: it assumes that the behaviour of the physical simulation is sufficiently similar to the real instrument used to record the actions, it requires skilled musicians and specialized equipment to record the musicians' actions, and it assumes that the musicians will always use the same actions to produce the desired sound rather than adjusting their actions on the fly.

In this work on performing with multiple distinct string quartets, we used physical modelling to create the audio. Sampling synthesis and even SMS are limited to the instruments available in the sample library. Recording a new instrument — such as adding a second distinct-sounding violin to a library with one existing recorded violin — is an expensive undertaking, requiring a recording studio, audio engineer, a skilled violinist, etc. One might consider altering the original instrument's audio to create a "new" instrument, such as slightly altering the pitch or filtering the upper partials, but these techniques are not perfect. By contrast, creating a new instrument with physical modelling is much simpler: measuring physical constants from a new instrument can be done quickly. One important factor of sample-based synthesis is that it does not reproduce the sound of a particular instrument; rather, it produces the the sound a particular instrument, being played by a particular musician, performing a particular set of pitches and timbres. When recording a single note on a piano, the musician has minimal effect, as the only interaction is the speed and force of depressing the piano key. But for continuous-excitation instruments such as the violin, clarinet, or human voice, the musician's control of the instrument plays a huge role in the resulting sound. Recording the sound of a skilled musician is a benefit when seeking a quick reproduction of decent sound, but it limits the flexibility of the music.

Physical modelling allows us to consider the instrument separately from the musician and performance style. To address the problem of controlling the physical models, we turned to machine learning to train the system to perform virtual instruments. Previous uses of machine learning in performance focused on recognizing performer gestures [12], while machine learning has been used off-line to evaluate beginning violinists [2], recognize cellists from their timbre [3], rank violins (rather than violinists) [36], and estimate physical actions from violin audio [25]. Active learning [4] is a technique in machine learning in which the computer generates examples for the user to evaluate; it has been used to generate personalized audio equalizers [21]. In contrast, we use a supervised learning approach to "teach" the virtual performer/listener to characterize the quality of the sound generated by the physical model and adjust the controls dynamically to achieve correct intonation and good timbre.

1.2 Our System: Vivi, the Virtual Violinist

Our goal is to allow a string player (not necessarily a professional) to train the system to perform any simulated violin, viola, or cello in a few hours with no specialized equipment. Once the system (Figure 2) is trained to recognize good or bad timbre, it automatically calibrates various parameters for each specific instrument. Given an electronic version of sheet music, the system extracts musical events to create a list of notes with extra parameters such as "note begins a slur" or "*glissando* from A4 to C♯5". These notes are fed into an intelligent feedback loop which adjusts low-level physical actions in response to the audio and haptic output of the physical model.

Our work bridges the gap between physical modelling of instruments, machine learning on acoustic signals, and musical performance. The main contributions of this paper are: 1) an intelligent feedback control system for violins trained with supervised and adaptive learning with no additional hardware required, 2) a flexible method of calibrating the system to perform a new bowed string instrument (violin, viola, cello), 3) a complete platform for the performance of string quartets, allowing researchers to work on one individual aspect of the process without requiring them to be experts in other areas. To enable and encourage future research, the entire system and all training data is available under a permissive copyleft license (the GNU GPLv3).

This paper is structured as follows: Section 2 gives a summary of the physical modelling library we used, while Section 3 details the machine learning and feedback loops involved in performing notes. Section 4 discusses the calibration of initial values used for the feedback control, and Section 5 describes the interpretation of a musical score into notes. Section 6 gives an overview of the video output, and Section 7 discusses the system and future work. Section 8 gives concluding remarks.

2. PHYSICAL MODELLING

We used the Artifastring library [22, 23], a highly optimized C++ library performing physical modelling of violin, viola, and cello. The model's string vibrations and bowing algorithm was based on Demoucron's work [9]: given the constants in Table 1, the string displacement $y(x,t)$ at position x and time t, subject to external forces $F_i(x,t)$, is:

$$\rho_{\mathrm{L}}\frac{\partial^2 y(x,t)}{\partial t^2} - T\frac{\partial^2 y(x,t)}{\partial x^2} + E\frac{\pi d^4}{64}\frac{\partial^4 y(x,t)}{\partial x^4} + R_{\mathrm{L}}(\omega)\frac{\partial y(x,t)}{\partial t} = \sum_i F_i(x,t) \quad (1)$$

The wave equation (1) is decomposed into the sum of modes with eigenvectors $\phi_n(x) = \sqrt{\frac{2}{L}}\sin\left(\frac{n\pi x}{L}\right)$. The bow friction is modelled with the classical hyperbolic friction curve (Equation 2) [20], with the addition of the stochastic term $\mu_e = \mu_c u(t)$, where $u(t)$ is a uniform random value $0.95 \leq u(t) \leq 1.0$. $\Delta v = v_0^h - v_{\mathrm{b}}$, with v_0^h being the velocity the string would move under the bow if there were no external forces.

$$F_{\mathrm{friction}} = \begin{cases} F_{\mathrm{b}}\left(\mu_d + \frac{(\mu_s-\mu_d)\mu_e}{\mu_e-\Delta v}\right) & \text{if } \Delta v < 0 \\ -F_{\mathrm{b}}\left(\mu_d + \frac{(\mu_s-\mu_d)\mu_e}{\mu_e+\Delta v}\right) & \text{if } \Delta v > 0 \end{cases} \quad (2)$$

For efficiency, Artifastring models the bow as a single point force. Modelling the bow as two point forces would require solving a pair of non-linear equations with two variables or a single quartic (polynomial of degree 4). As may be expected from a modal simulation, the resulting sound changes considerably if the bow position is a simply-rational position, meaning a proportion of the string away from the bridge with a small denominator, such as $\frac{1}{6}$ or $\frac{1}{7}$.

Symbol	Explanation
ρ_{L}	Linear density [kg/m]
T	Tension [N]
L	Length [m]
d	Diameter [m]
E	Young's Elastic Modulus
$R_{\mathrm{L}}(\omega)$	Frequency-specific damping
μ_s, μ_d, μ_c	Friction coefficients
s	String number [0,1,2,3]
x_b	Bow-bridge distance [normalized]
F_b	Bow force [N]
v_b	Bow velocity [m/s]
x_f	Left-hand finger position [normalized]

Table 1: Physical constants and control variables used. A body impulse from each instrument was also recorded.

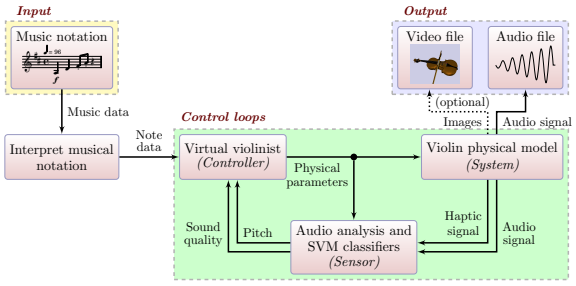

Figure 2: Overview of *Vivi, the Virtual Violinist.*

Finger forces (left-hand finger and right-hand plucking) were implemented as damped springs. The instrument body was modelled as a linear time-invariant system by convolving the sum of string outputs with 35 ms of a recorded impulse response. The library includes measurements of 5 violins, 2 violas, and 3 cellos. In addition to the audio output of the instrument body, the model also provides two signals for each string s: the audio output $A_s[t]$, and the right-hand haptic output arising from the force of the bow on the string $H_s[t]$. For further details, see [22].

3. INTELLIGENT FEEDBACK CONTROL

The mixed strength and curse of physical modelling is that it reproduces the real-world physics of instruments. When played by an expert musician, the results are very pleasing; however, as any parent or neighbour of a young child learning violin knows, when played by a novice the sounds can be rather harsh sounding. A violin is a complex system with many pitfalls which may not be apparent at first glance. For example, the pitch is not controlled exclusively from the left-hand finger placement: all actions of the bow (x_b, v_b, F_b) affect the pitch and can alter it by -70 to +20 cents[1] [28, 9]. Due to the non-linear friction equation, the string can react in a very different manner depending on those existing vibrations; furthermore, due to the stochastic $u(t)$ value, the string does not always react in the same manner. This is illustrated in Figure 3, and prompted us to use adaptive

[1]Cents are a logarithmic measurement system for musical pitch; given two pitches a and b, the cent $x = 1200 \cdot \log_2\left(\frac{b}{a}\right)$.

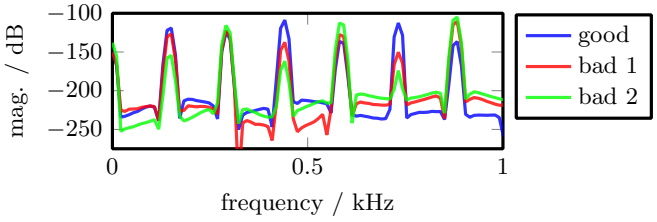

Figure 3: Different output with identical bowing parameters. Cello D string, bowed for 1 second with $x_b = 0.104$, $v_b = 0.5$ m/s, $F_b = 0.5$. The third example was created by bowing the string for 1 second with $v_b = -0.5$ m/s, then 1 more second after setting $v_b = 0.5$ m/s. There is a large different in the odd partials. Audio examples: `http://percival-music.ca/mm2013.html`

Note beginning	
p_s	Physical control variables as per Table 1
m_s	Target pitch (in MIDI)
k_b	Keep bow force (for slurs and ties)
k_e	Keep ears: do not reset pitch history (for ties)
y_b	Bow position along bow (for video output)
Note ending	
p_e	Physical control variables as per Table 1
m_e	Target pitch (in MIDI)
l_b	Lighten bow force (to change string cleanly)
k_v	Keep bow velocity (for slurs or "ring")

Table 2: Note modifiers: the musical score is separated into a series of notes, each of which is performed with the feedback control.

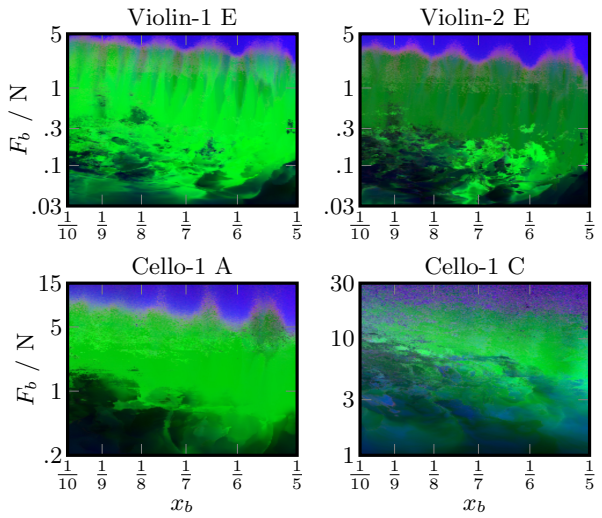

Figure 4: Schelleng diagrams of selected strings, the colour of each point indicating Red: standard deviation of spectral flatness measure (SFM); Green: mean of relative strength of f_0; Blue: mean of SFM. We desire a low standard deviation, strong f_0, and low SFM; so the best colour is light green.

feedback control rather than preset control values.

To accommodate the variability of our instruments, we used feedback control (Figure 2). There are two feedback loops: bow control (timbre, from the right hand) and finger control (pitch, from the left hand). The precise behaviour of the control loops receives minor modifications based on the type of note (shown in Table 2), but those modifiers only apply to the beginning and ending of each note.

Past work [27, 30, 14, 9] simulated string behaviour with constant input parameters (or in the case of [14], constant bow acceleration) mapping the region of parameters which results in the desirable Helmholtz motion in which the bow slips once per cycle of the fundamental frequency of the sound [15]. These plots can provide valuable insights into the behaviour of the model. Schelleng famously predicted the range of bow forces which could establish Helmholtz motion for a given bow velocity and bow-bridge distance [27]. Schelleng's predictions were later tested with an automated machine [29], where it was found that the predicted maximum bow force was quite accurate, while the minimum bow force differed from the predictions. Figure 4 shows "Schelleng diagrams" from empirical simulation. Each diagram was generated from one million simulations: 200 positions of x_b, 200 forces F_b, with each pair repeated 25 times to evaluate the stochastic behaviour. This exposes one problem of the modal simulation: on thin strings (e.g., violin E, cello A) the upper limit of F_b for good sound varies depending on whether x_b is close to a simply-rational position such as $\frac{1}{6}$.

Dynamic	Bow position x_b (fraction of L)	Bow velocity v_b (m/s)
ff	0.104	0.50
f	0.118	0.42
mf	0.134	0.34
mp	0.154	0.26
p	0.176	0.18
pp	0.192	0.10

Table 3: Physical parameters determined by dynamic. The velocity will be negative for an upbow.

To avoid the problematic bow positions x_b, we turned to real-world violin pedagogy [5], in which violinists are instructed to play in a specific bow position (termed "lanes on the Kreisler highway") for different dynamics. We take inspiration from another aspect of violin teaching, wherein the amount of bow for each note is often specified by the teacher (such as "half" or "quarter bow"), and the tempo is given by the piano accompaniment. Teachers sometimes direct students to imitate the teacher by playing music with the student (sometimes called the "mirror game"), in order to emphasize the amount of bow length to use. Students must therefore adjust their bow force to produce a good tone regardless of the teacher-imposed bow velocity. Acoustics research [28] showed that the overall loudness of a note is correlated with $\frac{v_b}{x_b}$. We set the bow velocity according to dynamic as well; the preset values for x_b and v_b are shown in Table 3.

3.1 Interactive Machine Learning

To train the computer to adjust the bow force F_b, we drew inspiration from human pedagogy: A human "teacher" will give judgements about simulated audio from the physical model. Those judgements are used to train a set of "virtual ears" which can then adjust the bow force in the absence of the teacher. This is analogous to young students practicing with the Suzuki violin method [34], in which a parent of the child is encouraged to aid in the home practice. Some poetic license is used here in calling the machine learning feedback "virtual ears". In addition to using the generated audio signal as input the supervise learning we also utilize a haptic signal from the bow's contact with the string. This

Class c	Human judgement of sound	F_b in example
1	not audible	0.002 N
2	"wispy" or "whistling"	0.11 N
3	acceptable	0.83 N
4	"harsh" or "detuned"	5.32 N
5	not recognizable as violin sound	10.9 N

Table 4: Human judgements of bowing. Audio examples: `http://percival-music.ca/mm2013.html`

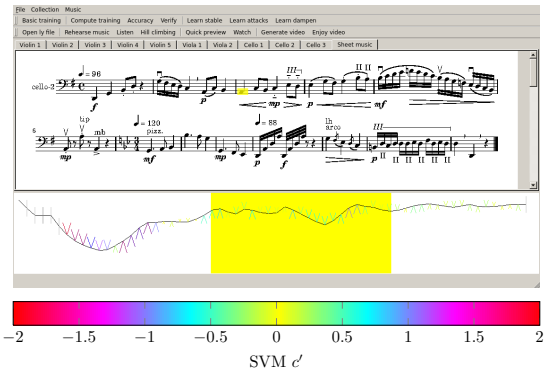

Figure 5: Screenshot of interactive training with our system. Middle portion of screen: the musical score with the first note in bar 3 highlighted. Bottom portion of the screen: shows the F_b used throughout the note (in black) and the SVM output c' (coloured arrows up or down); a portion of the SVM output has been selected (in yellow) for labelling and retraining.

String	Num.	String	Num.	String	Num.
Violin G	97	Viola C	59	Cello C	71
Violin D	67	Viola G	62	Cello G	68
Violin A	73	Viola D	54	Cello D	55
Violin E	56	Viola A	61	Cello A	99

Table 5: Number of examples in the datasets.

"virtual finger" provides additional information that can help produce and maintain a good tone, intended to mimic a real musician's sensations in her bow hand.

There are five possible judgements about the sound, presented in Table 4. The initial training of the system takes place within an interactive python script, where the user freely adjusts F_b in order to create audio examples for each class. Once the initial training is complete, the user listens to music while watching the sheet music, and indicates any incorrect judgements by clicking on the note(s) in the score and labelling the questionable audio as shown in Figure 5.

Internally, features are extracted from the audio $A_s[t]$ from the currently-bowed string s, and haptic $H_s[t]$ output from the bow (Section 2). A standard set of features are extracted from each signal using Marsyas [35]. Time-domain features used were RMS, peak to average ratio, and zero crossings. Spectral-domain features used were the rolloff, centroid, relative energy in the f_0 peak, flatness measure, and power to average ratio. These features were computed on both the $A_s[t]$ and $H_s[t]$ signals and concatenated together. Finally, x_f and x_b were appended to the feature vectors. The resulting vectors were used to train one Support Vector Machine (SVM) classifier for each string. The "virtual ears" operate at 100 Hz, with a sampling rate of 44100 Hz, hop size of 441 samples, and window size of 2048 samples. There are a total of 12 SVMs, one for each string of each instrument. Once fully trained with the musical exercises in Section 4.3, the SVMs are very accurate (99.5%–100% accuracy in 10-fold cross-validation). Each training example in the databases (Table 5) consist of three files: audio data, haptic data, and musician actions.

3.2 Bow Control: Timbre

While performing a note, the "virtual ears" are used to dynamically adjust the bow force F_b in the simulation in order to fix any unpleasant sound. Concretely, the $A_s[t]$ and $H_s[t]$ signals from the currently bowed string s are fed into relevant SVM, which outputs a probability $p_c[h]$ that the input vector belongs to class c for hop h. We define the weighted mean of these probabilities $c'[h] = 3 - \frac{1}{5}\sum_{c=1}^{5} cp_c[h]$ to be used to adjust the bow force:

$$F_{\mathrm{b}}[h+1] = \begin{cases} F_{\mathrm{b}}^i & \text{if } h < W \\ e^{\ln(F_{\mathrm{b}}[h]) - c'[h]K} & \text{else} \end{cases} \quad (3)$$

To avoid the initial transients which have little relation to the labelled audio from the sustained portion of the sound, we omit the first 30 ms (3 frames) from the control loop. We then begin filling the window, but avoid machine learning judgements until the window does not contain any uninitialized data (wait another 4 frames). We thus set $W = 7$. F_b^i and K are calibrated for each string, dynamic, and three pitches in Section 4. If the desired note parameters are not one of the calibrated values (e.g., a note in the middle of a *cresc.* or a different pitch), F_b^i and K are linearly interpolated from the nearest two calibrated values.

By default, notes are assumed to be détaché "on the string"; this is the first bow-stroke taught in the Suzuki method of violin playing [34]. In terms of the control loop, the bow force is non-zero before beginning to move the bow. The bow velocity v_b and position x_b are set based on the dynamics in Table 3. We experimented with setting v_b to 0 and accelerating to the target v_b, but this produced greater variability in the note attacks as the string would sometimes stabilize in undesirable states such as those shown in Figure 3. v_b is set to zero 50 ms before the ending of the note. For this final portion of the note, F_b is gradually reduced ($F_b(t+1) = 0.5 \cdot F_b(t)$) to avoid a sudden "pluck-like" sound which occurs if the bow force is suddenly set to zero.

The bow control loop is modified according to the following variables from Table 2. If k_b is enabled, then F_b^i is not used and $F_b(0)$ is retained from the final F_b in the previous note. If k_v is enabled, then then v_b is not set to zero at the ending of the note, and F_b is not reduced at the end of a note. In addition, F_b is not reduced if l_b is enabled.

3.3 Finger Control: Pitch

String players must adjust the finger position x_f to achieve correct intonation. Pitch detection was performed with the YINFFT algorithm, an extension [1] of the YIN algorithm [8]. The difference between the reference pitch and the output of YINFFT was taken modulo 12 to avoid octave errors (the variation in pitch due to the bow actions is less than 1 semitone). This difference was median filtered with a filter of length 3, becoming the error term $e[h]$, which is used in

$$x_f[h+1] = \begin{cases} x_f[t] + K_M e[h] & \text{if } |e(t)| \geq 0.01 \\ x_f[t] & \text{else} \end{cases} \quad (4)$$

$K_M = 0.01$ and the cutoff of 0.01 MIDI pitch units were set experimentally. The finger control is modified according to the following variables from Table 2. The reference pitch is always set according to m_s. If k_e is enabled, then the pitch filter buffer is not reset at the beginning of a new note.

3.4 Validating the Bow Controller

To verify that the feedback controller is useful, we performed a few tests comparing it with a naive set of actions (i.e. constant F_b). Past analysis of audio from violinists and cellists (i.e. focus on the performers) [2, 3] and violins (i.e. focus on the instrument) [36] used machine learning and statistical methods to distinguish good timbre from bad timbre, so we will do the same thing and rely on our "virtual ears" to provide judgements. Concretely, for each simulated note, we define the cost as the RMS of SVM judgements $c'[h]$ for all H frames. A cost of 0 indicates that the "virtual ears" found the note to be pleasing, while a cost of 2 indicates that the note had very bad timbre (either not audible or not recognizable as violin sound, as per Table 4).

$$\text{cost} = \sqrt{\frac{1}{H}\sum_{h=0}^{H}\left(c'[h]\right)^2} \tag{5}$$

Three cases will be tested: the feedback loop as described, the feedback loop without haptic output, and disabling the feedback (i.e. using a constant F_b). Although F_b^i and K are normally calibrated (Section 4.2), for these evaluations we will vary F_b^i and fix K. While the feedback is active, we set $K = 0.05$; to disable the feedback, we set $K = 0$. In all cases, the "virtual ears" will provide SVM judgements $c'[h]$, but when $K = 0$, (Equation 3) will have no effect on F_b which will effectively disable the feedback control. Without the haptic signal, the lowest 10-fold cross-validation accuracy falls from 99.5% to 98.4%.

Figure 6 shows the system performing two notes (without resetting the string in between) on the two "extreme" open strings in the string quartet (violin E is the highest-pitch string, cello C is the lowest-pitch string). Recall that the physical model contains a stochastic element, so multiple performances of the same note will produce different audio (as shown in Figure 3). The feedback control is quite flexible, showing a lower cost if a poor F_b^i is used. However, if a good F_b^i is used (the middle of the F_b^i range shown), then there is only a small difference between the cost of a constant bow force and the feedback control. This is no surprise; the control loop exists to correct poor F_b values, and an optimal F_b requires no modification. In many cases of an optimal F_b, the string will either fall into Helmholtz motion during the initial attack (as discussed in Section 3.2, we omit the first 30 ms from the "virtual ears"), or else will reach Helmholtz motion soon thereafter as the string's vibrations evolve naturally under even a constant F_b. The effect of removing the haptic signal from the machine learning is not noticeable for the first note bowed, but the cost increases for the second note bowed. In some cases, the cost is higher than the optimal F_b^i values. This occurs if the "virtual ears" mistakenly directs the controller to change F_b when there is a good tone, or to move F_b in the wrong direction, then the timbre will suffer.

Although the pair of simulated notes failed to show a compelling advantage of feedback control over a finely-tuned constant F_b^i, the true strength of feedback control is revealed in the longer musical examples in Figure 7. Our system is trained interactively with users, so we wish to minimize the calibration time after each set of retraining. As a result, we rarely (if ever) have an optimal F_b^i. This is particularly relevant for notes requiring an F_b^i interpolated from two calibrated values. As shown in Figure 8, the system only calibrates three pitches for each string. A scale (especially a chromatic scale) requires a great deal of F_b^i values estimated from interpolating between calibrated values. The feedback controller with no haptic signals performed quite well with the regular scale but poorly on the C string. Closer examination of the data from the scale revealed that the cello D and A strings were "easier to play", in the sense that they easily produced good timbre. The distinction in playing difficulty was suggested by the Schelleng diagrams in Figure 4.

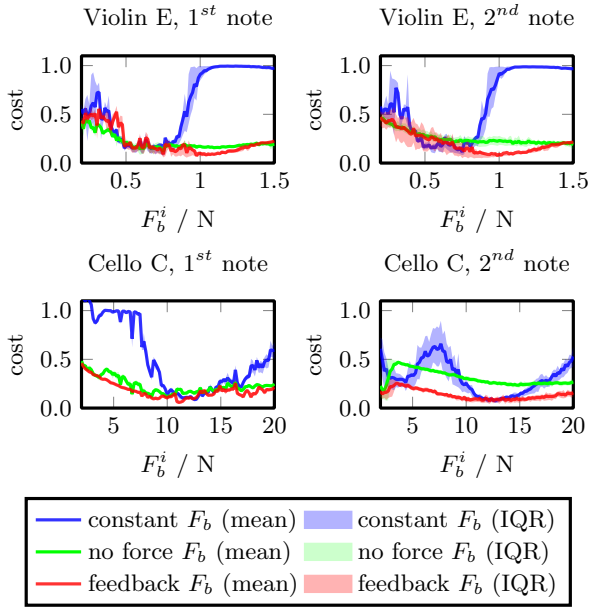

Figure 6: Comparison of naive vs. controlled performance of open strings *ff*, with the string initially at rest and then bowed for two notes. The range of costs arising from 50 simulations is shown by the inter-quartile range (IQR).

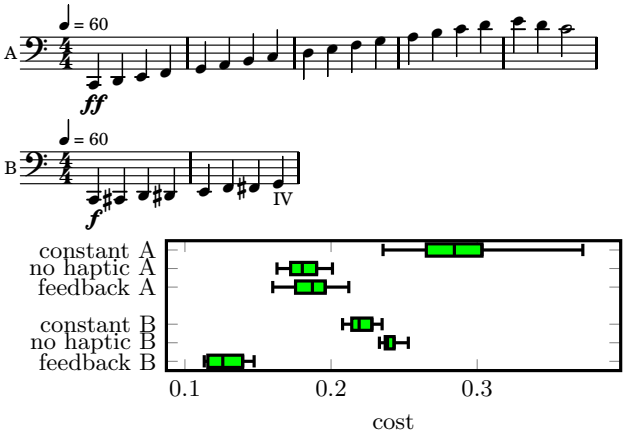

Figure 7: Comparison of naive vs. controlled performance of exercises, box plot of 10 simulations. Each box indicates the 25%, 50% (median), and 75% percentiles, while the whiskers indicate the minimum and maximum values.

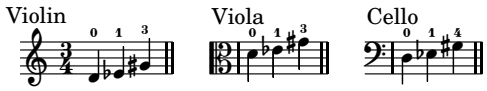

Figure 8: "Basic fingers" evaluated for training and calibration. Only one string for each instrument is shown; other strings used the same positions (i.e. 0, 1, and 6 semitones above the open string).

4. CALIBRATION

The feedback control requires two values: the initial bow force F_b^i and rate of change K. These are calibrated with the trained machine learning and by performing simulations. This calibration is performed during the training sessions (at a time chosen by the user), so we were required to balance the quality of the calibrated values against the time users spend waiting. Values were calibrated for three finger positions for each string-dynamic pair, shown in Figure 8, On a modern desktop computer with four simultaneous threads operating, calibrating one string takes approximately five minutes.

4.1 Checking the SVM Behaviour

The 10-fold cross-correlation accuracy is computed for each SVM, but this does not guarantee that the SVM is adequate for controlling the bow force. If the training set does not cover the entire range of signals produced by the physical model, the SVM could be 100% accurate on a proper subset but not useful for controlling the model.

To check the behaviour, a series of simulations are performed. Checking the validity requires human attention, but this can be done visually instead of listening to each audio sample. A display similar to Figure 9 is shown, showing the cost of notes with various F_b^i. A success "sanity test" is achieved if the extreme values of F_b^i on the left and right sides are red/dark and there exist some inner values in green/blue — this indicates that the range of F_b^i includes values both above and below the optimal value.

This process is also used to estimate the range of F_b to investigate in greater detail. It begins by using the maximum and minimum F_b values as F_b^i. After those two instances, it performs a binary search to find the boundary between the "too low", "too high", and "potentially good" F_b^i values. In particular, any simulation whose mean of c' is less than -0.1 is saved to a set of V_l; above 0.1 is V_h, while the remainder is V_m. At each step, two ratios are computed: $\frac{\min(V_m)}{\max(V_l)}$ and $\frac{\min(V_h)}{\max(V_m)}$, indicating the range between the low, middle, and high sets. The next exercise is computed with F_b^i being in the middle of the largest ratio. This is repeated until there are 9 examples.

4.2 Calibrating F_b^i and K

Once the SVM behaviour has been verified, we perform a grid search to find F_b^i and K (Figure 9. K is evaluated with 5 values linearly spaced from 0.0 to 0.2, while F_b^i is evaluated with 15 values linearly spaced from $\max(V_l)$ to $\min(V_h)$. Each (F_b^i, K) pair is evaluated by simulating 4 notes, all beginning from rest. As was done in Section 3.4, the cost of each pair is defined as the RMS of the c' judgements. An extremely high value of K can result in the system oscillating between F_b being too low and F_b being too high, particularly

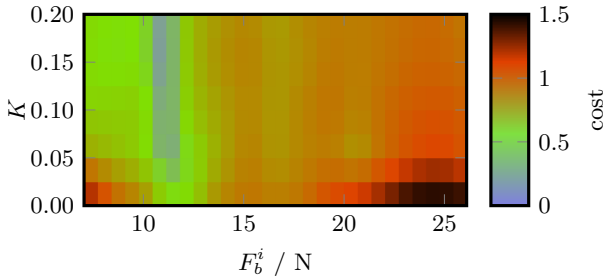

Figure 9: Calibrating Cello C, *ff*, open string

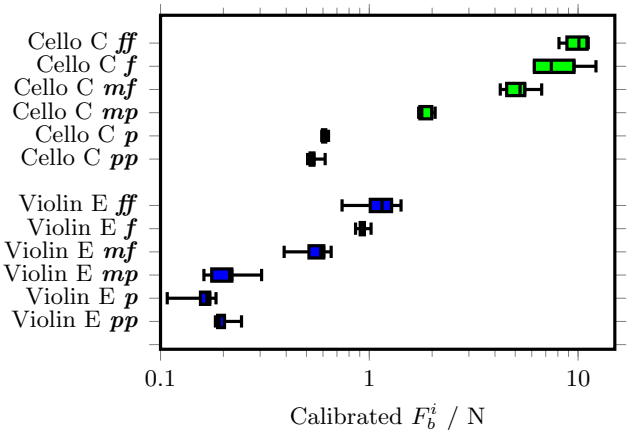

Figure 10: 10 calibrations of two open strings. Each box indicates the 25%, 50% (median), and 75% percentiles, while the whiskers indicate the minimum and maximum values.

since the analysis window is slightly longer than 4 frames.

Figure 10 shows multiple calibrations of two strings to demonstrate the reproducibility and usefulness of this step. There are clear differences in F_b^i for each string and dynamic.

4.3 Musical Exercises

After completing the initial calibration, the user is encouraged to listen to a number of musical exercises shown in Figure 11. This additional interactive training could be performed using any sheet music, but we prefer to use musical scales and exercises as they are traditionally a fundamental step in instrumental practice; as a musician, it "feels natural" to hear (and correct) mistakes in timbre by practicing scales. However, there are also solid technical reasons to practice scales before attempting more complicated music: it allows us to pinpoint and fix problems in isolation. Correcting mistakes in musical exercises involves a certain amount of repetition – identify a few mistakes, label them, retrain the SVM, then perform the exercise again. Retraining the SVM takes approximately five seconds provided that we skip the F_b^i stage, and simulating any of the exercises shown in Figure 11 takes less than one second. Longer music (naturally) takes more time to simulate, and longer to listen to. The musical exercises cover most of the potential problems which would be found in real music, so it is appropriate to ensure that the performance of these exercises is satisfactory before attempting more complicated sheet music.

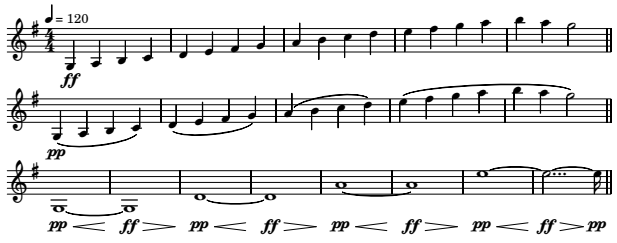

Figure 11: Musical exercises; the first two scales are performed with all six dynamics, while the *cresc.* / *decresc.* is performed as notated.

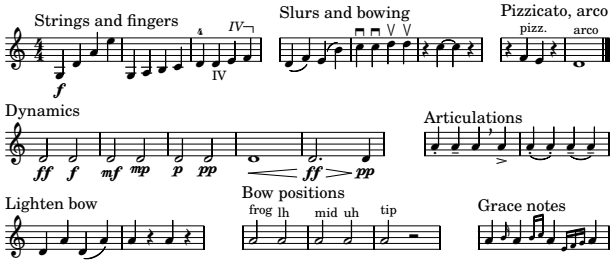

Figure 12: Musical notation understood

5. INTERPRETATION OF NOTATION

Our system interprets sheet music written for the LilyPond[2] music engraver. Music in MusicXML format can be converted to LilyPond using the `musicxml2ly` tool. Musical events (e.g., notes, slurs, dynamics) are extracted from LilyPond using scheme (a lisp variant). Figure 12 shows the musical notation which is supported. Once a list of music events has been obtained, they must be mapped onto the mid-level note modifiers described in Table 2.

Some musical notation has a clear meaning, but many elements of notation have no fixed meaning. The latter category is known amongst musicians as "interpretation", and is a matter of research and debate about historical performance practices, the composer's intentions, and/or the performer's desires. Since this paper focuses on the feedback control and interactive training of a string quartet, we avoided the question of interpretation by simply hard-coding values for ambiguous notation. Since the system was initially trained with the music in Suzuki violin books 1 and 2 [34], we chose constants which sounded best for Baroque and Classical music.

The system assumes that each note is played on the highest string s which can produce the desired pitch; in more musical terms, it assumes that all notes are in first position, other than higher-position notes on the E string. If the musical score indicates that notes should be played on a particular string with typical string player notation ("sul G", "IV", or a non-open fingering written over a note which could be played with an open string), that notation overrides the default behaviour. The finger position x_f and reference pitch m_s are set according to equal temperament. As discussed in Section 3, dynamics set x_b and v_b to the somewhat-arbitrary values specified in Table 3. Gradual changes of dynamics (*cresc.* or *decresc.*) are assumed to be a linear interpolation of x_b and v_b between the starting and ending dynamic. If the pitch is one of the "basic finger" positions, then F_b^i and K are set according to the parameters determined in Section 4. If the pitch is between 1 and 6 semitones above an open string, the parameters are linearly interpolated between those calibrated values. Finger positions higher than 6 semitones will have F_b^i and K set as the value calibrated for 6 semitones.

When two or more notes are connected with a slur, the bow direction does not change between notes. In particular, v_b does not alternate, and the interior notes have k_b and k_v enabled. Two notes connected with a tie receive the same treatment, with the addition that the second note of the enables k_e. The bow direction can be specified with downbow and upbow signs. These set v_b as indicated, but do not enable k_b or k_v. In other words, the bow velocity returns to 0 at the end of each note, and each note begins with a new F_b^i.

Staccato and *portato* articulations indicate that the note should be shorter, but their exact musical meaning varies based on historical period, composer, and even the musician's personal style. We arbitrarily set a *staccato* note's duration to be 0.7 times the original duration (followed by a rest which is 0.3 times the original duration); a *portato* note is 0.9 times the original duration (again followed by a rest). When these articulations occur within a slur, they behave similar to two downbows in a row — the bow direction does not change but the bow velocity decelerates to 0 at the end of each note. A breath mark sets the previous note to 0.5 times the original duration (followed by a rest). In addition, the note has the additional modifier k_v. An accent indicates that the note should be "stronger". In our system, this is interpreted as doubling that note's F_b^i.

On a real violin, playing closer to the frog results in a heavier bow (increased F_b), while playing closer to the tip results in a lighter bow. However, this aspect of violin mechanics is not included in the physical model, so these indications are purely cosmetic for the video generation. The notation "frog" sets the bow-contact point along the bow hair to 0.0; "lh" (lower half) is arbitrarily 0.2, "mid" is 0.5, "uh" (upper half) is 0.8, and "tip" is 1.0. Grace notes are implemented by "stealing" duration from the previous note. The duration of grace notes is fixed to be half of the notated duration; this is a reasonable approximation of the usual interpretation musicians use for grace notes.

6. VIDEO OUTPUT

In order to allow users to easily visualize the musical performances, we added computer animation using Blender[3], an open-source computer graphics suite which provides python bindings, allowing us to automatically convert any set of physical actions into computer animations. Using a physical model for the sound synthesis and animation allows the final product to blend seamlessly, in contrast to sampling approaches in which the animations are an entirely separate process from the sound generation. We only have one model (the violin in Figure 13); to create animations for other instruments, we use alternate camera angles as shown in Figure 1.

The bowing action places the bow on the string at position x_b, finding the location by linearly interpolating between the

[2] `http://lilypond.org`

[3] `http://www.blender.org`

Figure 13: Close-up of violin animation. The orange cones on the fingerboard indicate finger positions; this violin is playing a rolled A major chord. Video examples: `http://percival-music.ca/mm2013.html`

two ends of the relevant string. The contact point along the bow hair is found by linearly interpolating between the two ends of the bow hair according to the position along the bow hair given by the bow velocity.

The four strings are not oriented on a single plane; the two inner strings are farther away from the instrument body than the two outer strings. This displacement is necessary to allow the bow to play each string without touching the other strings. The bow hair must therefore be placed at a different angle for each string. For each inner strings, the angle is set to be parallel to the adjacent strings. For the two outer strings, those angles are increased.

The video for each instrument in a string quartet is generated independently, then the images are concatenated to copy the normal seating arrangement of a string quartet, with violin 1 being in the bottom-left position and cello in the bottom-right. The audio for each instrument is generated as a monophonic audio file, but are mixed together with the instruments being panned linearly from -0.5 to 0.5.

7. DISCUSSION AND FUTURE WORK

The music created by our system is not suitable for most commercial applications. While physical modelling has the potential to allow more expressive music than than sampling-based approaches, this comes at the cost of increased difficulty of control, requiring highly skilled (virtual) musicians to achieve high-quality musical performances. Professional musicians require thousands of hours of deliberate practice to learn their craft [11]. The feedback control in our system does not completely replicate those years of training; the final audio contains occasional harsh attacks and squeaks, in a manner similar to the sounds produced by human students with a few years of training. This may be a boon for composers writing for student musicians, but most practical applications of synthesis would benefit from alternate methods rather than our system at the present time.

We plan to quantify the perceived quality of our music. We will perform a double-blind listening test, where participants will judge two or more performances of the same musical work. Participants will be asked to rank each performance on two axes: the quality of the performance, and the degree to which the performance is "human-like". One performance will be from our system, another from a commercial notation software package, and additional copies from any research projects which can create string quartet music. If the latter is not possible, then we will compare music from solo instruments rather than quartets. We will use established metrics such as the Friedman test and the Wilcoxon signed-rank test for evaluating the significance of our findings. In case the listener's musical background is a factor, we will conduct the survey with two groups: string players, and the general public.

There are a number of ways that we hope to improve the control loops. The present system relies on users with some amount of musical skill ($\approx$ 1 year of violin lessons) to judge whether the bow force F_b should be increased or decreased. We would like to remove even this small amount of background knowledge. Ideally, users would simply judge whether the audio was good or bad, and the system would determine the corrections required via more sophisticated machine learning which track time or state such as Hidden Markov Models and/or further calibration routines.

An alternate approach would be to deliberately imitate specific performances; this follows real-life violin pedagogy. Suzuki violin students are explicitly given audio recordings so that they can learn from them by attempting to imitate the recordings' sound. In addition to audio timbre distance metrics, such a system could involve video to estimate bow velocity or even bow-bridge distance. This type of approach was used by VocaListener and VocaWatcher [13] to create performances with Vocaloid singing synthesis [17] and a humanoid robot which matched a recording of a human singer.

The present system focuses on the low-level control, and used hard-coded values for many expressive musical decisions such as the length of *staccato* notes. Future work will provide a GUI to allow users to easily modify such stylistic decisions, allowing them to personalize the audio performances. There is an active research community focused on expressive computer music performances for piano music [16], but in the immediate future our goal is to give humans better/easier control over such decisions.

Finally, we will add vibrato (an oscillation of the left-hand finger). This is a skill taught to musicians with 3–4 years of experience, and will likely require some musical analysis to suggest which note(s) should receive vibrato and which notes should be played "straight".

8. CONCLUSION

We have developed a method of synthesizing audio and animated video from musical notation. Our method uses no recorded samples from musicians, instead using physical modelling and supervised machine learning to generate audio. This allows new instruments (e.g., a second distinct-sounding violin) to be added to the system with minimal effort. The machine learning and calibration routines are sufficiently flexible to apply to 4 violins, 2 violas, and 2 cellos with no manual tuning of constants.

To foster future research in this area, all code and data is available[4] under permissive copyleft licenses.

9. ACKNOWLEDGEMENTS

Many thanks to Marcos Press for creating the violin and bow models in Blender, and releasing it under the GPLv3.

[4] `http://percival-music.ca/vivi.html`

10. REFERENCES

[1] P. Brossier. *Automatic Annotation of Musical Audio for Interactive Applications*. PhD thesis, Queen Mary University, London, 2006.

[2] J. Charles, D. Fitzgerald, and E. Coyle. Violin sound quality detection. In *Irish Signals and Systems Conference*, 2008.

[3] M. Chudy and S. Dixon. Recognizing Cello Performers Using Timbre Models. Technical Report EECSRR-12-01, Queen Mary Uni., London, 2012.

[4] D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2), 1994.

[5] C. D. Collins. *Connecting Science and the Musical Arts in Teaching Tone Quality: Integrating Helmholtz Motion and Master Violin Teachers' Pedagogies*. PhD thesis, George Mason University, 2009.

[6] P. R. Cook. A Hierarchical System for Controlling Synthesis by Physical Modeling. In *International Computer Music Conference*, Banff, Canada, 1995.

[7] R. B. Dannenberg and I. Derenyi. Combining instrument and performance models for high-quality music synthesis. *J. of New Music Res.*, 27(3), 1998.

[8] A. de Cheveigné and H. Kawahara. YIN, a fundamental frequency estimator for speech and music. *J. of the Acoustical Society of America*, 111(4), 2002.

[9] M. Demoucron. *On the control of virtual violins: Physical modelling and control of bowed string instruments*. PhD thesis, IRCAM, Paris, 2008.

[10] J. W. Dunn, D. Byrd, M. Notess, J. Riley, and R. Scherle. Variations2: retrieving and using music in an academic setting. *Comm. ACM*, 49(8), 2006.

[11] K. A. Ericsson, R. T. Krampe, and C. Tesch-Römer. The role of deliberate practice in the acquisition of expert performance. *Psychological Rev.*, 100(3), 1993.

[12] R. Fiebrink, P. R. Cook, and D. Trueman. Human model evaluation in interactive supervised learning. In *CHI '11*, 2011.

[13] M. Goto, T. Nakano, S. Kajita, Y. Matsusaka, S. Nakaoka, and K. Yokoi. VocaListener and VocaWatcher: Imitating a human singer by using signal processing. In *ICASSP*, 2012.

[14] K. Guettler. On the creation of the helmholtz motion in bowed strings. *Acta Acustica united with Acustica*, 88(6):970–985, 2002.

[15] H. L. F. Helmholtz. *On the sensations of tone as a physiological basis for the theory of music*. Longmans, Green, London, 3rd edition, 1895. A. J. Ellis (transl.).

[16] H. Katayose, M. Hashida, G. De Poli, and K. Hirata. On Evaluating Systems for Generating Expressive Music Performance: the Rencon Experience. *J. of New Music Research*, 41(4), 2012.

[17] H. Kenmochi and H. Ohshita. VOCALOID – Commercial singing synthesizer based on sample concatenation. In *Interspeech*, 2007.

[18] E. Lindemann. Music Synthesis with Reconstructive Phrase Modeling. *Signal Processing Magazine, IEEE*, 24(2), 2007.

[19] E. Maestre. *Modeling instrumental gestures: an analysis/synthesis framework for violin bowing*. PhD thesis, Universitat Pompeu Fabra, 2009.

[20] M. E. McIntyre, R. Schumacher, and J. Woodhouse. On the oscillations of musical instruments. *J. of the Acoustical Society of America*, 74(5), 1983.

[21] B. Pardo, D. Little, and D. Gergle. Building a personalized audio equalizer interface with transfer learning and active learning. In *ACM MIRUM*, 2012.

[22] G. Percival. *Physical Modelling meets Machine Learning: Performing Music with a Virtual String Ensemble*. PhD thesis, University of Glasgow, 2013.

[23] G. Percival, N. Bailey, and G. Tzanetakis. Physical Modeling meets Machine Learning: Teaching Bow Control to a Virtual Violinist. In *Sound and Music Conference*, 2011.

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

[25] A. Perez Carrillo and M. M. Wanderley. Learning and extraction of violin instrumental controls from audio signal. In *ACM MIRUM*, 2012.

[26] G. P. Scavone and P. R. Cook. RtMidi, RtAudio, and a synthesis toolkit (STK) update. *ICMC*, 2005.

[27] J. C. Schelleng. The bowed string and the player. *J. of the Acoustical Society of America*, 53(1), 1973.

[28] E. Schoonderwaldt. The violinist's sound palette: spectral centroid, pitch flattening and anomalous low frequencies. *Acta Acustica united with Acustica*, 95(5), 2009.

[29] E. Schoonderwaldt, K. Guettler, and A. Askenfelt. Schelleng in retrospect – a systematic study of bow force limits for bowed violin strings. In *ISMA*, 2007.

[30] S. Serafin, J. O. Smith III, and J. Woodhouse. An investigation of the impact of torsion waves and friction characteristics on the playability of virtual bowed strings. In *Applications of Signal Processing to Audio and Acoustics, 1999 IEEE Workshop on*, 1999.

[31] X. Serra and J. Smith. Spectral modeling synthesis: A sound analysis/synthesis system based on a deterministic plus stochastic decomposition. *Computer Music Journal*, 14(4):12–24, 1990.

[32] J. O. Smith. Physical Modeling Using Digital Waveguides. *Computer Music Journal*, 16(4), 1992.

[33] J. O. Smith. *Physical Audio Signal Processing*. online book, accessed 2012-Jan, 2010.

[34] S. Suzuki. *Violin Part Volume 1*. Summy-Birchard Music, 1978.

[35] G. Tzanetakis. Marsyas: a case study in implementing Music Information Retrieval Systems. In S. Shen and L. Cui, editors, *Intelligent Music Information Systems: Tools and Methodologies*. Information Science Reference, 2007.

[36] P. Wrzeciono and K. Marasek. Violin Sound Quality: Expert Judgements and Objective Measurements. In Z. Ras and A. Wieczorkowska, editors, *Advances in Music Information Retrieval*, volume 274. Springer Berlin / Heidelberg, 2010.

[37] J. Yin, Y. Wang, and D. Hsu. Digital violin tutor: an integrated system for beginning violin learners. In *ACM Multimedia*, 2005.

[38] D. Young. Wireless sensor system for measurement of violin bowing parameters. In *Stockholm Music Acoustics Conference*, 2003.