First Steps Towards Violin Performance Extraction using Genetic Programming

Robert Scott Wilson

CCRMA,Stanford University,Stanford, California 94305,
rswilson@ccrma.stanford.edu,http://www-ccrma.stanford.edu/~rswilson/

Abstract

This paper outlines the outstanding problem of performance extraction from an audio signal of violin performance. It describes some preliminary programming runs that solve components of the larger problem and suggest that genetic programming is a viable aproach to solving the larger problem. Techniques for dealing with fitness cases and evaluation of time based signals using GP are presented.

1 Introduction and Overview

Ongoing work in the field of music acoustics has been focused on creating computation physical models of traditional musical instruments. The two primary uses of these models are in simulation and sound synthesis. As a simulation tool the models provide a base for further exploration and understanding of the dynamics of the musical instrument systems in question. As a synthesis tool a model's output may be and often is used as sound source material by composers and musicians in hardware and software music systems.

The missing piece between those two areas remains the problem of performance extraction. The problem of extracting control parameters for such a physical model from a real performance remains largely unsolved. Having the ability to intuit these parameters from a performance has been often persued goal in the field of computer music for decades. For pedagogical resasons the preformance transcription possibilities are enticing, but the application we are more focused on is obtaining a source of more true to reality parameter inputs for the physical models. These realistic inputs would be of great assistance in refining and validating the models as well as expanding the quality of sound synthesis achieved using them.

This paper presents the results of some first attempts at using genetic programming to evlolve programs that analyze audio signals and provide as outputs the necessary parameters to drive the digital waveguide model of a bowed violin.

Section 2 provides some background on other approaches to solving the problem performance extraction problem, other published uses of GP in processing time domain signal, and a brief overview of the function of the violin model in use in this study.

2 Background

2.1 Performance Extraction / Music Transcription

The problem of transcribing music from an acoustic signal into human readable musical scores has been extensively studied but remains largely unsolved except in very simple cases. In this project the goal is not to transcribe to human readable form which requires higher level music knowledge, but rather to find the

control parameters for a physical simulation of the instrument. When discovered these parameters can be considered a form of extreme data compression as the entropy in these control signals will be significantly lower than that of the original. The MPEG-4[SAOL] standard provides facility for this type of representation, but not yet the means to translate signals into this form.

Past efforts at musical transcription have made use of fourier transform representations, sinusiodal model representations, and/or statistical representations. Many are formulated as pitch detection problems in the presence of noise or competing signals. More recent work tends to draw from many of these domains concurrently and are starting to exploit the kinds of mathematical / computational models for the sources as we are using in this project.[Martin, 1996, Scheirer, 1995, Chafe]

2.2 Time domain signal processing using GP

Esparcia-Alcazar and Sharman[Esparcia-Alcazar]report some of the most promising results using GP for digital signal processing problems. In their work they have solved several channel equalization and interference removal problems using a combination of GP and simulated annealing. Another insteresting approach to time based signal processing using GP is the R-STROGANOFF [Iba] system which also uses GP to evolve a structure and a neural network style error propagation method to adjust weights at the nodes. These nodes are then used as the coefficients of polynomial functions to create the output. The drawback of this system over the above the Esparcia-Alcazar method seems to be in inplementation complexity, as the R-STROGANOFF requires that the state of the last output of any node in the tree be accessible as an argument to any other node in the tree.

Garcia[Garcia]reports success in using a genetic algorithm based pitch detection method to detect multiple simultaneous sounding pitches. No indication is given as to his treatment in time of the material.

2.3 The Violin Model

The digital waveguide bowed violin model uses a combination of three basic elements. Digital delay lines model wave propagation (velocity waves in this case) from the wave equation. Digital filters are used to model absorbtion and transmission at the bridge as well as the radiation and resonance properties of the body of the instrument. A nonlinear friction function is used to model the friction in the interaction between the bow and the string. The inputs to the nonlinear function are the bow force, differential velocity (between the string and the bow) and string slope.

Violinists apply rosin to the horsehair bow to give it some stickiness. When the player bows the instrument the string sticks to the bow for most of one period of the signal and then suddenly slips and initiates a flyback motion to return towards its resting position (where the potential energy and string tension are minimized) before sticking to the bow again and repeating the process. This stick slip process sets up a Helmholtz motion on the string as a velocity corner travels from the bow up to the rigid termination at the nut end of the string, reflects, and returns past the bow again to reflect at the bridge end before returning to the point of origin.

This ideal helmholtz motion only occurs in a certain region of the parameter space characterized by bow velocity accross the string, bow force down onto the string, bow position along the string, and the pitch of the sounding note. Figure 1 shows examples of the string displacement in the model under several different conditions. Waveform a) shows good stable helmholtz motion with one slip per period, waveform b) shows a multiple slip which can be indicative of insufficient bow force, and waveform c) shows the chaotic motion that can result for instance when the bow force is too strong.

We have implemented detectors inside the model to report whether it is operating inside this region or not and to estimate the sounding pitch based on the number of samples between successive flyback events for use as feedback to the GP program.

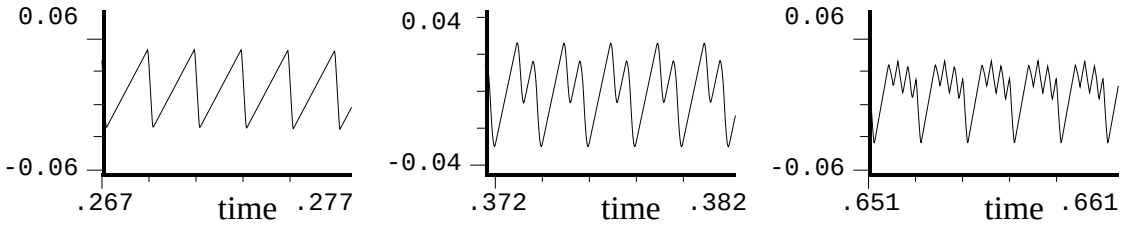

Figure 1: Helmholtz String Displacement. a) Ideal motion. b) multiple slips. c) chaotic motion

3 Methods

The study was implemented using the lilGP[lilGP] software for the genetic programming support and using a C++ violin model[Serafin] implemented in the Synthesis ToolKit[STK] and based on theory published sumarized in [Smith] which encapsulates a great deal of knowledge from the musical acoustics field. The simulations were run on single processor 1.5GHz athlon linux workstation and on an apple G4 powerbook under Macintosh OS X. A typical run of the evolution took between eight and twelve hours to converge.

Table 1 outlines the parameters fore a typical GP run on the violin problem.

3.1 Wrapper, Environment, and Pre-Processing of Fitness Cases

One of the biggest challenges in the work was coming up with a wrapper environment to present the time domain signal to the GP program in a useful but computationaly efficient way. A major design goal was to present both time domain and frequency domain representations of then signal to the program. As a result a block oriented approach was adopted for the sake of simplicity in generating and maintaining the frequency domain representations of the signal. Alternatives to the block scheme considered were a sample by sample approach on the one hand and a complete random access approach on the other.

The sample by sample approach would be preferable because a single time sample and the corresponding FFT of the signal centered at that sample could be presented at the input of the GP program for every iteration. Due to computational constraints this approach is not feasible. The computational load in this case is greater than the block method by the block size (1024 times greater in this case).

At the other end of the spectrum of design choices would be one that provides the GP program random access to the time and frequency data for the input and lets the program use it freely. The first problem with this approach is that of correlating the time and frequency inputs in a meaningful way. A second problem with such an approach is how to deal with input signals of varying lengnths. The third problem is again one of computational limitations as the size of the search space for this case is enormous. The population size and fitness evaluation costs would both be extremely large to solve this problem.

The block method used in this study takes a logical middle ground between these two extremes by presenting a full block of the time domain signal (1024 samples in this case) and the lower half of the frequency magnitude block. Only half of the FFT output is required as the FFT magnitude of any real signal is even symmetric and the phase component is odd symmetric. A variant of the block approach is uses blocks with an overlap factor of 50% in an effort to improve time resolution. In practice in a limited number of runs this approach did not seem to help, but rather complicated the task of the program.

In addition to the block by block FFT analyses, several other signal metrics are computed at the outset of a run for each of the .WAV file fitness cases. These metrics included a pitch estimate, the autocorrelation of the block, the energy and the bias of the signal. In this case the pitch detection algorithm used was the

Table 1: Tableau for a typical run of the gpviolin.

Objective:	Output FREQ, BOWVEL, BOWFORCE parameters for the digital waveguide violin physical model such that the error between the model output and the fitness case audio signal is minimized.
Terminal Set:	*ADF0*: ARG0 ARG1 *RPB*: TWO_PI, ERC_F, ERC_I, (wrapper inputs:) SAMPLE_IDX, PEAK1, PEAK2, PEAK3, FC_PITCH_EST, (model feedback terminals:) LAST_OUT, HELM_STAT, HELM_FREQ_EST
Function Set:	ADF0: *, /, +, -, SIN, COS, RLOG, REXP RPB: *, /, +, -, SIN, COS, RLOG, REXP, INPUT_TIME_GET, INPUT_FREQ_MAG_GET, INPUT_FREQ_PHASE_GET, DELAY_INTERNAL, DELAY_GET, DELAY_SET, ADF0
Fitness cases:	A combination of of short (1-2 second) audio clips of bach solo violin partitas, and violin notes generated by the physical model.
Raw fitness:	The total sum of weighted sums of several fitness measures for each frame analyzed. The fitness measures are MSE between model output and reference fitness case for magnitude frequency spectrum, autocorrelation function and time domain signal, plus 1.0 - mean output of the helmholtz motion detector.
Standardized fitness:	Raw fitness / number of fitness cases
Hits:	Number of frames who's individual raw fitness is < 0.01
Wrapper:	Preprocessing of fitness cases and application of generated parameters to a run of the violin model. (See 3.1 for details)
Parameters:	M = 500, G = 200, NUM_FITNESS_CASES = 4, FRAMES_TO_EVALUATE = 10, RANDOM_FRAME_START = 1, SAMPLE_RATE = 44100, DSP_BUFFER_SIZE = 1024, DSP_ANALYSIS_SIZE = 4096, FITNESS_VALUE_CUTOFF = 1024.0
Success predicate:	Hits = NUM_FITNESS_CASES * FRAMES_TO_EVALUATE

Harmonic Product Spectrum (HPS)[de la Cuadra].

3.2 Delay functions

Digital delay elements are a crucial aspect of this type of physical simulation. The structure of this GP setup is such that in the course of evaluating the program's fitness on a given sample the GP program is executed once for each sample. Given that the model we are deriving parameters for contains delay elements and has a response over time and that we are trying to model a highly correlated signal, giving the GP program mechanisms to maintain internal state over the course of these per sample runs is essential. This need for state is addressed in two ways, first the delay element functions that were developed for this purpose will be described, and next the feedback terminals from the model which solve the state problem in a complementary way will be discussed.

Three types of delay functions were created, DELAY_INTERNAL holds its value only until the next time it is read, DELAY always serves up the value that was stored in it on the previous run of the tree, and DLINE allows values to be get and set at any point with it, these values are shifted over one index in the array after each execution of the tree. The internal delay serves a slighly different purpose than the other two, its value was also carried over from one run to the next, but was more often used as a method of passing state between the three result producing branches of the tree. DELAY_INTERNAL does not have separate get and set functions as the other two delay types do, but simply always takes an argument and returns the value that it was last called with. This internal delay was also effective in other configurations which allowed iteration. In conventional signal processing terms DLINE is a single delay line of a fixed length with possible taps at every sample while the DELAY element was like a bank of single sample delay elements.

The author was able to evolve a digital waveguide resonator program in GP using only two DELAY elements, ephemeral random floating point constants and the +,-,* functions. This is as expected as the digital waveguide resonator can be simply modeled as two single sample delay elements recursively connected which cross feed each other through summing connections and two gain multipliers.

This set of elements is very simple implement but seems to serve the same function as much more complex schemes that keep track of node outputs throughout the program tree and permit arbitrary access to their last ouputs as described in [Iba]. The function set described is believed to allow access to the entire search space of parameter estimation for sample iterative processing as in this study. Iteration and conditional execution were tried on some runs but did not yield significant improvement on those runs.

3.3 Feedback Terminals

A very important set of terminals provided to the program were the feedback outputs from within the violin model. In addition to providing the model's previous output LAST_OUT as a terminal, values of the two detectors internal to the model, HELM_STATUS and HELM_FREQ_EST were provided.

4 Results

Graphical representations are presented for two individuals selected out of different GP runs. The output of the individual in Figure 2 is from the best of run individual at generation 15 in a population of 300 given three fitness cases. Two of these fitness cases were single note excerpts of a real violin recording, and the third was the output of a note generated by the same violin model. The experimental output (top pair) and the fitness case (bottom pair) can be seen to be very similar. The waveforms exhibit very similar shape except for the corner on the waveform which appears more in phase with the fundamental frequency in the recording than in the model output. The spectrogram excerpt at the right which shows frequency versus time with frequency bands of higher amplitude shaded darker than bands with less amplitude. The patterns

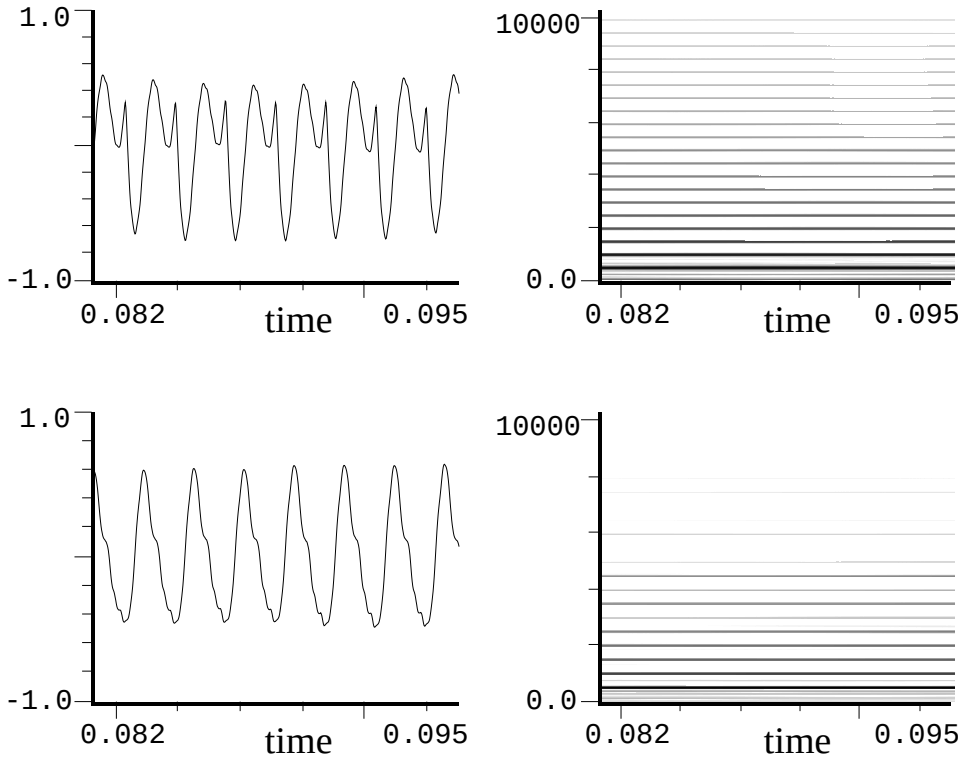

Figure 2: Waveform and Spectrogram of fit individual. Top: Model output, Bottom: Fitness case in question.

are very similar, showing that the experimental output matched the pitch and spectral content very closely. The spectral energy in the model output extends up much higher than in the case of the recording but this is due to the reverberation in the room and recording, as well as probable differences in the body of the recorded instrument versus that modeled in the simulation. This individual had a raw fitness of 0.258, standardized fitness of 0.086, and had scored 9 of a possible 63 hits (21 frames per fitness case were evaluate in this case).

Figure 3 show the three parameter outputs over time for the same individual and fitness case. In this case the program used the pitch detector input terminal directly as its frequency output parameter, it used derived the force from a combination of the pitch detector output and the phase response array, and the velocity was a large complicated function including all of the feedback and fitness case data terminals.

Figure 4shows the time and frequency domain outputs for a very unfit individual with maximum raw fitness and standardized fitness of 0.993. As can be seen there is much more spectral energy where there should not be. There are also abrupt discontinuities in the force and velocity outputs which as causing audible noise and contributing to this spread spectrum energy. In this case the frequency output was one of the interger ephemeral random constants (220) which is totally meaningless as the the expected values are in radians from 0.0 to π corresponsing to physical frequencies of 0Hz to 22050Hz. In practice the model only produces any meaningful output in the range of 50Hz to ~2kHz so values outside this range are cut off at those boundaries. This cutoff was helpful in avoiding numerically problematic situations as when operated with extreme parameters it tended to produce Inf and NaN outputs and complicate the fitness evaluation.

The code that follows is the output of another unrelated best of run individual from a population of 800

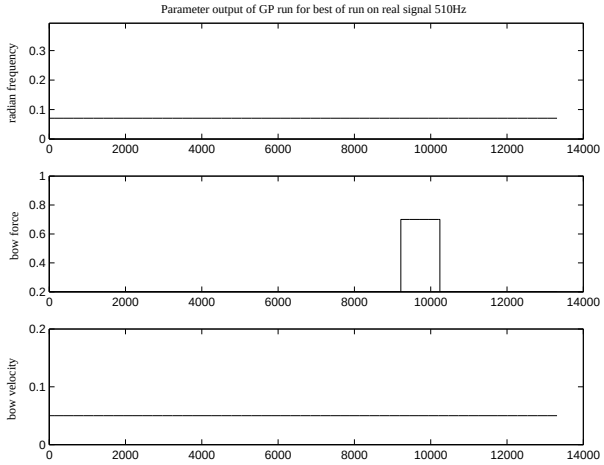

Figure 3: Parameters ouput for fit individual

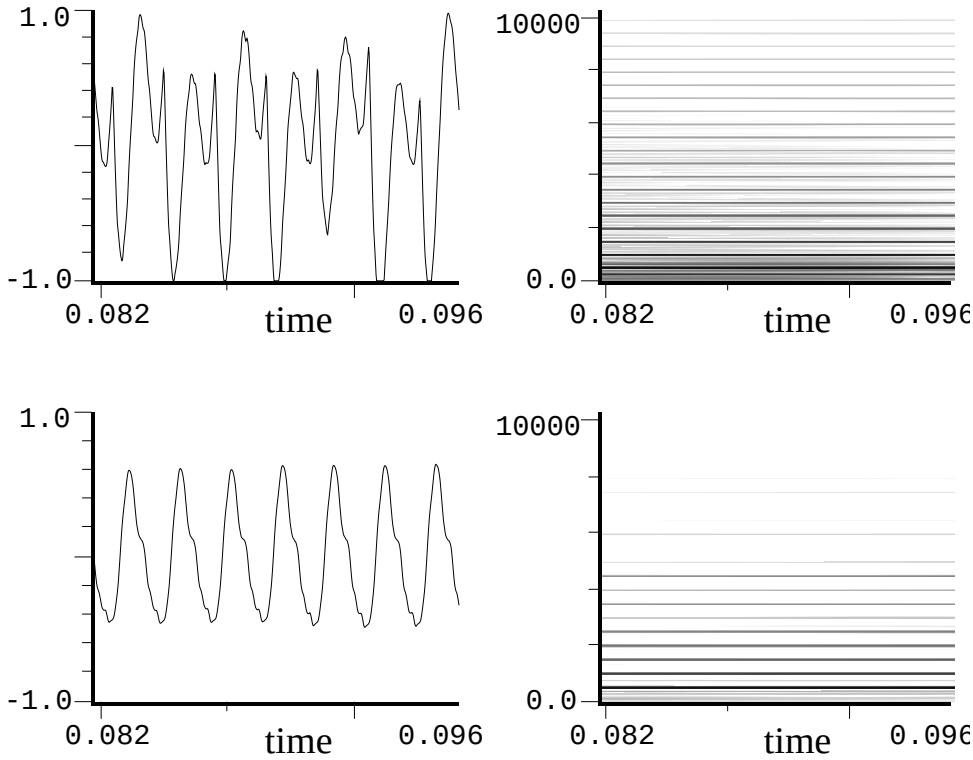

Figure 4: Waveform and spectrogram of highly unfit individual. Top: Model output, Bottom: Fitness case in question.

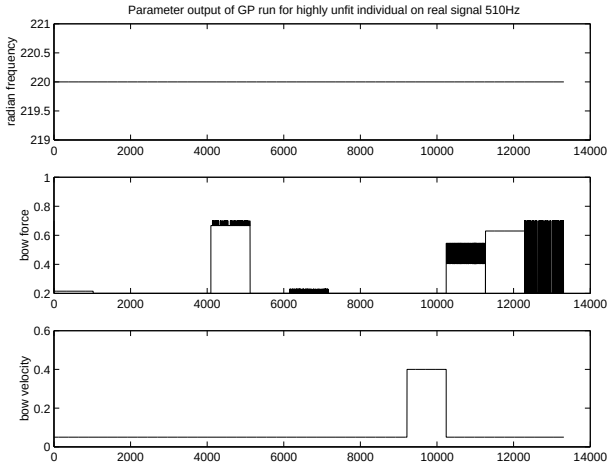

Figure 5: Parameter outputs for highly unfit individual.

where 8 fitness cases were provided, two of which were the real violin recordings. 8 frames per fitness case were evaluated in the fitness evaluation in this case:

```
FREQ: FC_PITCH_EST
FORCE: (input_time_get (/ (input_freq_mag_get FC_PITCH_EST) (* (/
(sin (* FC_PITCH_EST LAST_OUT)) (- HELM_STAT FC_PITCH_EST)) (sin
(sin (sin (sin (sin (sin (sin (sin (sin HELM_STAT))))))))))))
VEL: (sin (sin (sin (sin (sin (sin (sin (sin (sin
FC_PITCH_EST)))))))))
ADF0: (rlog (/ (- (+ 967 ARG1) (sin ARG1)) (sin (+ (exp (rlog (sin
(exp (cos (/ (* (* 0.74095 (cos ARG1)) ARG0) (sin ARG1))))))) (cos
(sin (rlog (/ (exp ARG0) (* ARG0 (sin (- (+ 967 ARG1) (sin (exp (sin
(cos (* (* ARG1 TWO_PI) (rlog 0.55129)))))))))))))))))
```

This individual was chosen for it's small code size, but is interesting because the force branch is looking up the spectral peak corresponding to the the fitness case pitch estimate and using it in a calculation. The ADF goes unused in this case.

Discussion of Results

These results demonstrate the ability of GP runs to accurately set frequency, bow velocity and bow force parameters of the violin in response to audio signals generated both from the model itself and from real world performance of a violin. Clearly the setting of frequency is insignificant given that a reasonably accurate pitch detection of the signal is provided as an input. Significant effort was put into trying to evolve programs that would correctly set pitch without being given this input, but it became clear that, while certainly possible, the range of fitness cases, complexity of programs, and sheer volume of computation required to achieve this result to a satisfactory level (ie. more accurate than the HPS pitch detector I had at my disposal) would have been impossible given the time constraints of the project. In fact on some of the runs made with a larger number of fitness cases and changes in pitch the frequency branch was not always simply the pitch detector output, but contained some more subtle calculation.

Having chosen to give the output of the pitch detector directly as a terminal seems reasonable given that this would almost certainly be an input to a practically used performance extraction system. The choice was made instead to focus on the setting of the combination of bow velocity and bow force within the correct playability region. This is more useful because the combionations of these parameters are not a known quantity with known algorithms. The two dimensional playability space of the model has a generally known shape[Serafin et al, 1999] which has been calculated by exhaustive simulation, but the region changes for every pitch on the instrument. This is due to many factors including the coupling of the strings through the bridge to the body resonator who's resonant peaks remain fixed despite playing different fundamental frequencies on the instrument.

One significant problem that is not resolved yet in that the human ear is extremely sensitive to the smallest perturbations or discontinuities in an audio signal. As a result the fitness cases must somehow take this into account which they don't at this time. This can be seen in the output in Figure 2 which contains some periodic modulation at a rate many times lower than the sounding pitch of the note and which hardly shows up in the fitness measures, but is not there in the fitness case sample. It is not surpising that the amplitude modulation distortion is not identified by the current fitness measures in part due to the blocking chose. That is less than single period of the low frequency modulation might occur over the course of a block of samples, but that distortion is clearly audible to the listener.

There is a visible artifact in the parameter output of Figure 3in the second half of the sample which is due the the limited number of frames that can be processed for each fitness case on each individual. On that run the frames computed for fitness all came at the beginning after one frame of warmup to let the model stabilize, so the region of the sample where the anomaly is was never tested during evolution. Some late runs used a randomized starting frame within the fitness case sample, but no conclusive results were available at the time of writing. Ideally we could randomly sample within the fitness case to test fitness, but this is impossible due to the nature of the model which (like the real instrument) stores energy over a short period of time and takes in the area of ten to twenty periods to stabilize its output.

5 Conclusion

This paper has demonstrated some successful small first steps towards building a performance extraction system for the digital waveguide violin model. The variety and number of fitness cases as well as well the population sizes used in this study are clealy far too small to be able to solve the performance extraction problem fully, indeed most of the runs in the study failed to ever converge completely. This is in part due to the inherent nature of the problem whereby we don't know if the model at hand is capable of exactly reproducing the real world inputs. However for the very focused subset of the larger parameter space, the velocity and force parameter pair for single note samples in a restricted frequency range, the model was successful. Best of generation individuals were always playing the model in the desired playability region and seemed to a certain degree to be exploiting the more subtle choice of these parameters within that playability region to closely approximate the timbral characteristics of the reference samples.

6 Future Work

There are many future directions this work will need to take on the road to a complete solution. More fitness cases which are both more widely varied and carefully tuned will be needed. To succeed on longer examples a scheme to provide feedback on several timescales at once is necessary. This would address problems such as the low frequency amplitude modulation mentioned above. The model will have to be complemented with an adjustable room model to compensate for the reverberation and recording style used in the input signal. The computational cost of performing thorough fitness testing for very large populations in this type

of problem is still prohibitive, but becoming less so rapidly.

References

[Serafin et al, 1999] Serafin, S., Smith, J.O., Woodhouse, J. An Investigation of the Impact of Torsion Waves and Friction Characteristics on the Playability of Virtual Bowed Strings. *Proceedings of the 1999 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics.* New Paltz, New York.

[Serafin] Serafin, Violin model implementation.

[de la Cuadra] de la Cuadra, P., Master, A. S., Sapp, C., Efficient Pitch Detection Techniques for Interactive Music. *Proceedings of the Internation Computer Music Conference 2001.* Havana, Cuba.

[Koza] Koza, J. R., Rice, J. P. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* Cambridge, MA: The MIT Press.

[Esparcia-Alcazar] Esparcia-Alcazar, A.I., Sharman, K.C. 1995. Evolving Digital Signal Processing Algorithms by Genetic Programming. *Technical Report CSC-95012*, Faculty of Engineering, University of Glasgow.

[Iba] Iba, H., Sato, T., de Garis, H., Temporal Data Processing Using Genetic Programming. *Proceedings of the Sixth International Conference on Genetic Algorithms*, 1995, University of Pittsburgh.

[Garcia] Garcia, G. A Genetic Search Technique for Polyphonic Pitch Detection. *Proceedings of the Internation Computer Music Conference 2001.* Havana, Cuba.

[Smith] Smith, Julius. Digital Waveguide Modeling of Musical Instruments `http://www-ccrma.stanford.edu/~jos/waveguide/waveguide.html`

[Martin, 1996] Martin, K., "A blackboard system for automatic transcription of simple polyphonic music". Perceptual Computing Technical Report #385, MIT Media Lab, July 1996. http://citeseer.nj.nec.com/keith96blackboard.html

[Scheirer, 1995] Scheirer, E.d., Extracting expressive performance information from recorded music. Master's thesis, Program in Media Arts and Science, Massachusetts Institute of Technology, 1995. http://citeseer.nj.nec.com/scheirer95extracting.html

[Chafe] Chris Chafe and DavidJaffe. Source Separation and Note Identifi- cation in Polyphonic Music. Proceedings of the IEEE Conference on Acoustics Speech and Signal Processing, Tokyo (2): pp. 25.6.1- 25.6.4, 1986.

[STK] The Synthesis Tool Kit. Princeton, Stanford, 2002.STK `http://www-ccrma.stanford.edu/software/stk/`

[lilGP] lil-gp Genetic Programming System `http://garage.cps.msu.edu/software/lil-gp/lilgp-index.html`

[SAOL] MPEG-4 Structure Audio Orchestra Language `http://sound.media.mit.edu/mpeg4`