Violin Driven Synthesis from Spectral Models

Greg Kellum

Master thesis submitted in partial fulfillment of the requirements for the degree:

Master in Information, Communication, and Audiovisual Media Technologies

Supervisor: Xavier Serra

Department of Information and Communication Technologies
Universitat Pompeu Fabra
Spain
September 2007

Table of Contents

Abstract

The aim of this project is to explore the use of the violin as a controller for real time synthesis. In this document acoustical features unique to bowed strings are identified which are relevant for controlling sound synthesis. Algorithms are given for extracting these features using both time-domain and spectral domain-based techniques. And a real-time synthesizer is presented that uses these feature descriptors to control a sample-based synthesizer. In particular an application scenario is presented where a violin is used to control synthesis from a spectral model of a guitar being played with an ebow.

Acknowledgements

During my stay at the MTG, I had the luck to be immersed in an environment which encourages unique synergies by bringing together many researchers with different research interests to work together side by side on projects in the field of music technology. I am grateful to all of the members of the group for making the most of this opportunity through their openness, supportiveness, and generosity towards one another.

In particular I would like to thank Xavier Serra, my supervisor, for incorporating me into the group and for the direction he gave to this project. I would like to thank Alfonso Perez for his advice regarding violin acoustics. I would like to thank Pau Arumi and the other members of the CLAM project for their help in realizing a spectral synthesis application using the CLAM framework, and I would like to thank Jordi Bonada for his advice about how to overcome many of the practical problems one encounters when creating a real-time spectral modeling synthesis application. Finally, I would like to thank Google, Inc. for their financial support.

Barcelona, Spain Greg Kellum
September 3, 2007

I. Introduction

This work was motivated by the author's frustration with existing tools for composing music using audio samples. Having spent several years using applications such as AudioSculpt, Melodyne, RTCmix, and CSound for transforming samples in atypical manners, the author became increasingly dissatisfied with the drawbacks of working with off-line sample manipulation applications. In particular such applications are time intensive. They do not allow one to easily try out new ideas or variations on old ideas. They constrain one's ability to make global changes in a piece with regards to key or tempo. And they simply are not very fun to work with; the process of making music with such applications is in no way comparable to playing a musical instrument.

Nonetheless, such applications are very powerful. For those readers who are unfamiliar with any of the aforementioned applications, put simply, they expose more of the content of audio samples so that users can manipulate samples in a manner befitting their content. They allow one to isolate instantaneous moments of a sound and extend them or to isolate particular aspects of a sound and strip away other non-essential parts. They allow one to build new composite sounds from constituent parts by adding, morphing, filtering, or convolving one sound with another. In short they allow one to sculpt sounds that one has imagined from other sounds that one has found manifested in reality.

For this reason the author began to look for ways to bring the power of these techniques into real-time by looking on one hand for controllers that offer more degrees of freedom than typical MIDI controllers and on the other hand for real-time synthesis techniques that allow one to manipulate samples based on their content. After a couple of years of experiments with alternative controllers on one hand such as virtual reality gloves, joysticks, augmented percussion instruments, etc. and granular synthesizers and phase vocoders on the other hand (Kellum,

2005), the author came to the idea underlying the project described in this document: to use one of the most expressive musical instruments – the violin – as a controller for one of the most powerful sample modeling techniques – sinusoidal plus residual modeling.

Fortunately, there have been other projects of a similar nature, which the author was able to learn a great deal from and which have influenced the direction of this project.

In 1998 Wessel et al presented a real time audio transformation technique, which allows audio material to be reconstituted with an arbitrary pitch, duration, and spectral evolution given control inputs of pitch and volume. After performing a sinusoidal analysis of some performances, Wessel used various machine-learning techniques, e.g. neural networks, to learn the relationships between control parameters and analysis frames so that when given a set of control inputs during a performance the best fitting analysis frame could be selected for resynthesis regardless of its temporal position in the analysis file.

Around the year 2000, the company, Antares, released an audio-driven synthesizer named Kantos. Antares also makes the popular pitch correction software, Autotune, and Kantos presumably uses the same underlying pitch tracking algorithm. Although Kantos' pitch tracker occasionally makes octave errors and is not safe in the face of unexpected noise and other transients, it nonetheless works surprisingly well, i.e. well enough to use in a concert setting. The results of the monophonic pitch analysis are used to control a wave-table based synthesizer. This type of synthesizer loops a short segment of audio material indefinitely, varying the pitch and amplitude of the loop in response to changes in the control inputs. The sound produced by such a synthesizer is oftentimes so uniform that it can quickly become uninteresting, and the fact that Kantos never was successful as a product is likely due to this rather primitive synthesizer. Kantos has been discontinued by Antares.

Zeta, the well known maker of electric violins, has also created a hardware based, audio driven synthesizer for use with their violins named

the Synthony II MIDI Processor. According to the product literature, it is a sample based synthesizer capable of playing instrument sounds including drums, horns, woodwinds, pianos, guitars, basses, and special effects. Yoo and Fujinaga (1998) evaluated the Synthony II by comparing it to two other pitch trackers, and they mentioned only that it had the highest latency of the pitch trackers they tested.

The project with perhaps the single most relevance to ours was Tristan Jehan's master project done in collaboration with Bernd Schoner at MIT's Media Lab (Jehan, 2001). Jehan developed an audio driven synthesizer, which extracts perceptual information from an audio stream including pitch, loudness and brightness and uses it to control a spectral model based synthesizer. Although Jehan's system was designed to be usable with an arbitrary audio source, it was showcased in a performance where it was played by a violinist. Jehan's system is notable for its use of sophisticated statistical techniques in mapping the control source to the synthesis engine. Jehan makes use of a probabilistic inference framework, cluster weighted modeling (Schoner, 2000), to predict the timbre of the synthesizer from the control inputs. Jehan's work was certainly an advance, but some users have found his system to be difficult to control in a truly perceptually meaningful manner. When used with a violin, the brightness values extracted by his system seem to be little better than noise, and the pitch tracker does not handle transients very well. These problems derive from Jehan's underlying assumption that the system should be usable with an arbitrary audio source. A model of brightness or pitch that works well with one instrument will not necessarily work well with another, and by trying to accommodate an arbitrary audio source, one is lead to use feature descriptors that cater to the lowest common denominator of audio features. However, in all fairness, Jehan mentioned his intention to develop new and better audio descriptors for his system in the conclusion of his thesis and regardless of whether or not he has done so, he certainly recognized that improving his audio descriptors would improve the performance of his system.

That concludes our review of the literature related to audio-driven synthesis. In the first section of this document a review of the literature on violin acoustics is made in which the acoustical properties of the violin are presented which are relevant to audio driven synthesis. In the following section methods are presented for extracting the pitch, amplitude, brightness and bow direction of the signal as well as for classifying individual frames as stationeries or transients. Within this section the subsection on pitch detection evaluates the efficacy of the existing Yin algorithm (Cheveigne, 2001) for estimating the pitch of the bowed strings while the subsections on brightness estimation and stationary / transient classification present primarily new methods developed by the author. Subsequently, a real-time spectral modeling synthesis application for continuous control sources is presented. This synthesizer uses the existing CLAM implementation of spectral modeling synthesis (Amatriain et al, 2006) but adds some important extensions to this framework, which are necessary for creating a general musical instrument.

II. Overview of the Acoustics of the Violin

The violin is the most wonderful of instruments, because it possesses more subtleties of color and shading than all other instruments.
Rimsky Korsakov

In the following section on violin acoustics, the manner in which bowed strings vibrate is described. The relationship between bowing parameters and the spectral attributes of the produced sound wave is discussed, and the transition that bowed strings undergo to arrive at steady state motion from rest is explicated.

A. The effect of bowing parameters on tone

The violin is one of the most difficult musical instruments to understand in terms of acoustics. A bowed string is constantly losing energy through its dissipation into sound and heat, yet the bow is constantly providing additional energy, which ideally serves to keep the string vibrating in a stable manner. Small changes in the manner in which the violin is bowed can lead to sudden and unexpected changes in the manner in which the string vibrates causing the string no longer to vibrate in a stable manner. This marks the bowed string as a non-linear system, and non-linear systems have traditionally been difficult to model (Woodhouse and Galluzzo, 2004).

This fact partially explains why it takes so many years to learn to play violin with any great proficiency. One of the skills that violinists acquire over years of practice is the ability to consistently induce *Helmholtz* motion in the string despite changes in bow pressure and bow-bridge distance. (The concept of "bow pressure" is also oftentimes referred to as "bow force" within the literature on violin acoustics. I prefer the term "bow pressure" over "bow force" and will use it consistently throughout this essay.) Although a vibrating violin string appears to the

naked eye to move in the same manner as a vibrating rubber band, i.e. the entire string appears to move back and forth as a single arc, when viewed with time-lapse photography or examined using other techniques, Hermann von Helmholtz discovered in the late 19th century that in reality in any given instant the string forms a triangle with the bow. Which is to say, that the string divides into two parts, which meet at a peak called the Helmholtz corner, and over time this peak runs up and down the length of the string. What one perceives as an arc-like shape is in reality the envelope of the movement of the Helmholtz corner as it traverses the string.

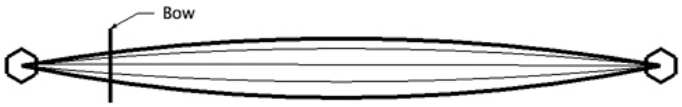
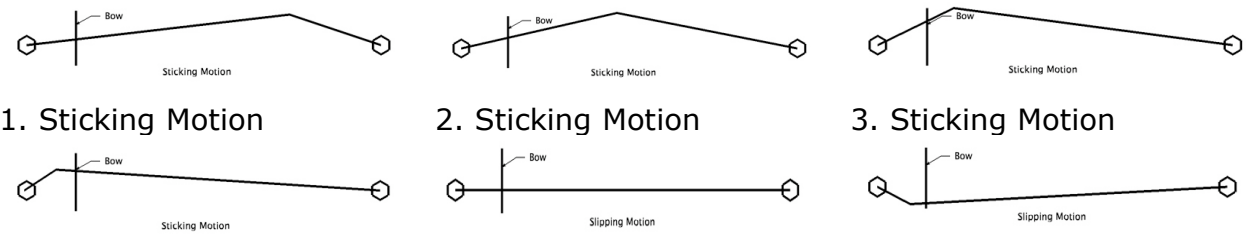


Figure 1 Bowed string motion as it appears to the naked eye

When the Helmholtz corner is moving from the violinist's finger towards the bow, the string sticks to the bow; the friction between the string and the rosin on the bow causes the string to be dragged by the bow. But when the Helmholtz corner crosses the bow and moves towards the bridge, the string slips and moves in the opposite direction of the bow. The alternation between these two types of motion constitutes Helmholtz motion, which is the type of motion that violinists almost exclusively sought to achieve before the birth of modern classical music.



1. Sticking Motion
2. Sticking Motion
3. Sticking Motion
4. Transition to Slipping Motion
5. Slipping Motion
6. Slipping Motion

Figure 2 Bowed string in Helmholtz motion

There are, however, other manners in which the string may move. If the violinist uses very little bow pressure when playing, two Helmholtz corners can form rather than one, causing a type of motion referred to as double-slipping motion. The resulting sound wave has a different waveform from the wave produced by Helmholtz motion, but it possesses the same pitch. As a wave produced in this manner lacks the energy to fully excite the resonances in the body of the violin, the resulting sound wave sounds hollow and uninteresting, and for this reason violin teachers train their students to avoid bowing in this manner.

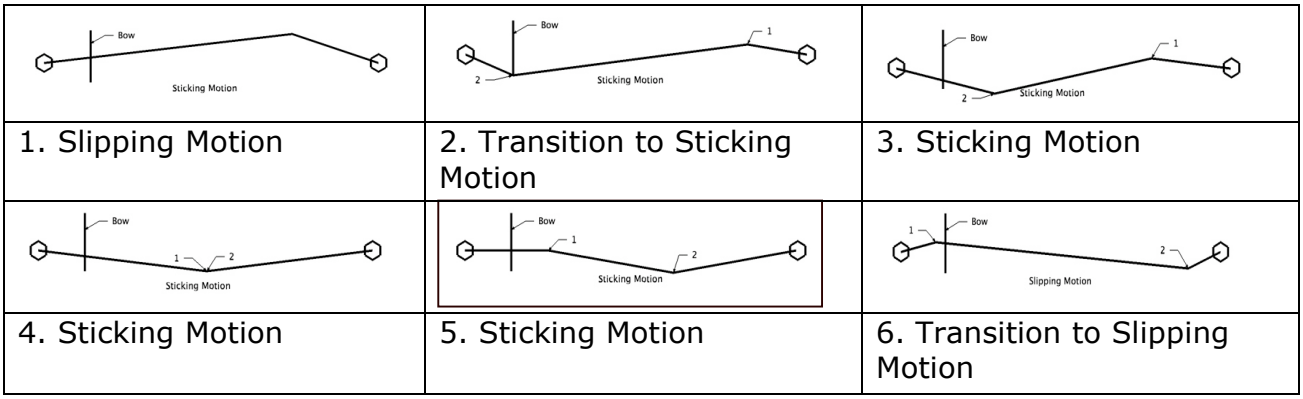


Figure 3 Bowed string in double sticking motion

On the flip side of the coin, it is also possible to deviate from Helmholtz motion by applying too much bow pressure as well. In this case the string oftentimes sticks to the bow even when the Helmholtz corner crosses the bow while moving towards the bridge. The resulting sound is extremely rough and noisy and borders on aperiodicity.

The amount of bow pressure alone, however, does not fully determine whether a string will settle into Helmholtz motion or one of the other alternatives. The distance of the bow from the bridge of the violin is also a deciding factor. It has been shown if a string is length L and the distance of the bow from the bridge is βL, then the maximum allowable bow pressure to achieve Helmholtz motion is proportional to β^{-1} and the minimum allowable bow pressure is proportional to β^{-2}. As originally suggested by John Schelling, this state of affairs can be represented

schematically on a logarithmic scale as follows:

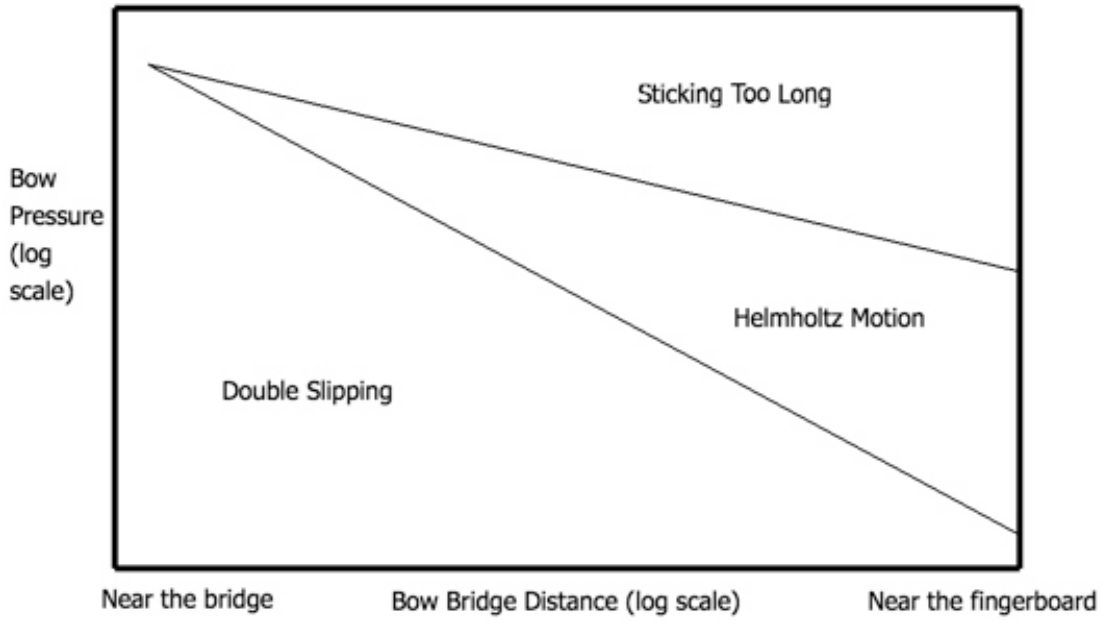


Figure 4 Schelling diagram displaying the range of possible bow pressures and bow bridge distances that produce Helmholtz motion.

The violinist may control the brightness and the amplitude of a violin tone independently (within bounds) by his choice of bow pressure, bow bridge distance, and bow velocity (Askenfeld 1986). Increasing the bow pressure for instance while holding the other parameters constant produces a brighter sound. This increase in upper frequency content results from the fact that as the Helmholtz corner travels up and down the string it is rounded off as it approaches the string's end points. And when the corner passes under the bow again before moving towards the finger board, it is resharpened, the extent of the resharpening depending on the amount of bow pressure being applied (Cremer, 1984); the sharper the corner, the stronger the higher partials will be in the resulting sound wave. This brightening of the tone will also cause it to be perceived as louder although the amplitude of the sound wave is essentially the same, due to the fact that the human auditory system is more sensitive to higher frequencies. Decreasing the bow bridge distance will also in and of itself make the tone brighter, because the spectrum of

the produced sound wave has a continuous number of partials up to an order which is proportional to the ratio between the length of the string and the bow-bridge distance (Benade, 1990). Oftentimes, these two effects combine, because as one decreases the bow bridge distance, one must necessarily increase the bow pressure to maintain Helmholtz motion. And therefore, as a general rule, one may say that bowing closer to the bridge will increase the brightness of the tone. The amplitude of the produced sound wave on the other hand is principally affected by the velocity of the bow. The amplitude principally derives from the distance that the string is pulled by the bow, and the extent of this distance is directly proportional to the bow's velocity. The bow bridge distance, however, plays a role in determining the range of possible amplitude values, as the maximum possible amplitude of a string's vibration decreases as this distance grows (Helmholtz, 1885). These relationships, known as Helmholtz's classical steady-state theory, can be formally stated as follows. The peak displacement amplitude $\hat{u}$ at a point x along a string with length L and fundamental period T_0 is given by:

$$\widehat{u} = \frac{T_0(L-x)}{2L^2}\frac{v_B}{\beta}, \quad x < \frac{L}{2}$$

where $\beta = (x_B / L)$ is the fractional distance from the bridge to the bowing point. As one can see from this equation, $\hat{u}$ is proportional to the ratio v_B / β. When understood in physical terms, v_B / β corresponds to the step in relative velocity between bow and string when switching between sticking and slipping (Askenfelt, 1988) This mathematical formulation merely serves to expand upon, however, what was already previously elucidated: that the bow velocity and bow bridge distance are the violinists main control over the peak amplitude.

For the sake of completeness, let us briefly look at two further parameters that a violinist has at his disposal: the bow position, i.e. the point of contact between the bow and the string, and bow tilt. Both of

these parameters affect the range of bow pressures that a violinist may apply. It is only when the bow position is close to the frog that the violinist may exert the maximal bow pressure whereas the minimal bow pressure can be applied most safely near the tip of the bow. Tilting the bow similarly affects the range of possible bow pressures by reducing the width of the contact area between the bow hairs and the string from 10 mm to only a few millimeters; this reduced contact area allows violinists to play with very little bow pressure (Askenfelt, 1988). Both of these parameters can, therefore, be used to affect the brightness of the tone produced by the violin.

B. Transients

In the previous section we discussed the motion of a string when it is in a steady state, but before the string arrives at a steady state, there is a transient period where the motion of the string is typically aperiodic. This transient period can vary in duration and character depending on the style of bowing used as well as other factors, but three categories of transients predominate: (1) Periodic slipping of the string from the very beginning, giving periods equal or close to the period length of the Helmholtz motion (2) Multiple slips, where more than one slipping interval occurs during each fundamental period and (3) Prolonged irregular periods characterized by raucous sounds oftentimes with no clearly definable pitch (Guettler, 2002). The first category of transition proceeds to steady-state motion as follows:

When the bow first starts to move across the string, the string is pulled outwards. The string then slips, and two waves radiate outward from the bow. The first slip ends after the second wave passes the bow and moves towards the fingerboard. These waves both then reflect off of the finger and move in the direction of the bridge. The first wave has the wrong sign to cause slipping, and it instead reflects off the bow rather

than the bridge. The second wave causes the bow to begin slipping which continues until it has been reflected off of the bridge and once again crosses the bow. As the second wave traveled a longer distance than the first, these two waves are now farther apart. These waves then continue to move in this manner until one of the waves eventually overtakes the other, and the string settles into Helmholtz motion (Woodhouse, 2004).

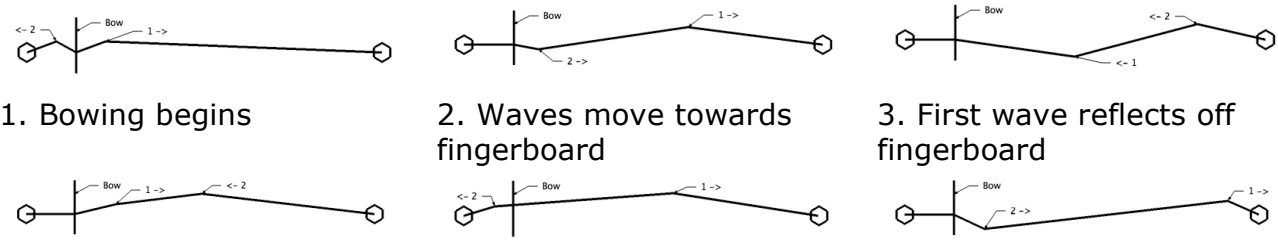


Figure 5 Bowed string in transient phase

This manner of transient can last as little as 5 milliseconds before the string settles into steady Helmholtz motion. Surveys have shown, however, that transients of this nature can last up to 50 milliseconds and still be deemed acceptable by professional violinists (Guettler, 2002). The second category of transition – multiple slips – can last up to 100 milliseconds with the approbation of professional violinists, but professional violinists have much less patience with the third category of raucous transients.

III. Feature Extraction from the Violin's Sound

In this section algorithms are presented for extracting the pitch, amplitude, brightness, and bow direction of the violin's signal as well as for classifying whether a given frame value is a stationary or transient.

A. Pitch detection algorithm

Pitch is that attribute of auditory sensation in terms of which sounds may be ordered on a scale extending from low to high (ANSI 1973).

Pitch tracking is an area of music technology that has been worked on by countless researchers, but nonetheless, research in this area continues unabated with new articles being published every few years detailing recent advances. Part of the reason for this continued activity is the ambiguity inherent to the concept of pitch itself. Although the ANSI definition given above seems reasonable for a great number of sounds, there are more difficult cases where a sound can possess formants that suggest a pitch other than that of the sound's period, and when confronted with such sounds, some listeners may indicate the periodic pitch to be "the pitch," while other listeners will indicate the formant's pitch (also known as the *spectral pitch* in Terhardt 1974). In the light of this and other difficult cases, pitch has been revealed to be a multidimensional concept. And as a result, when one speaks of pitch tracking algorithms, one must be clear about what model of pitch one has in mind, because different pitch tracking algorithms are underlain by different simplifying models of pitch.

With regards to the violin, one can safely say that the period of the signal yields the pitch. This implies that either the time-domain technique of auto-correlation or spectral domain techniques employing pattern matching would be suitable for extracting the violin's pitch. The author

began by performing an informal survey of these techniques using Alain Cheveigne's Yin time-domain autocorrelation method (Cheveigne et al, 2001), Tristan Jehan's spectral-domain maximum likelihood estimator (which was derived from Miller Puckette's work) (Jehan, 2001), and Maher and Beauchamp's spectral-domain two-way mismatch algorithm (Maher and Beauchamp, 1993). This survey showed that the spectral domain techniques had a much higher error rate for the violin than auto-correlation. This was likely due to the fact that the violin has strong resonances in the body, which made it difficult for the spectral domain techniques to identify the spectral peaks, which are harmonics of the fundamental. In addition to this informal survey, a much more comprehensive evaluation of various pitch tracking algorithms was performed by the author of the Yin algorithm, Alain Cheveigne, using a database of speech recorded together with a laryngograph signal, and this survey showed Yin to have error rates that were three times lower than competing methods (Cheveigne, 2001). The auto-correlation method also offers the additional advantage of providing a value for the aperiodicity of the signal, which is useful for identifying transients that occurred during bow changes. So, for these reasons the author undertook to develop a real-time version of the Yin algorithm in C++ using Cheveigne's existing, publicly available matlab version as a reference.

1. The Yin algorithm

Yin is classified as an auto-correlation algorithm. Auto-correlation is a method of finding the period of a signal by multiplying the signal with time-shifted versions of itself. The autocorrelation function of a discrete signal x_t may be defined as

$$r_t(\tau) = \sum_{j=t+1}^{t+W} x_j x_{j+\tau}$$

where r_t(t) is the autocorrelation function of lag τ calculated at time index *t*, and *W* is the integration window size. For a perfectly periodic signal, the autocorrelation function has a maximum at that lag, i.e. time-shift, of the signal which corresponds to the period of the signal.

Yin does not actually use the auto-correlation function, but rather a function from the same family – the squared difference function.

$$d_t(\tau) = \sum_{j=1}^{W} (x_j - x_{j+\tau})^2$$

Here we search for the smallest lag τ for which the function is zero, as a perfectly periodic signal will always be zero when offset by its period or by a multiple of its period.

The squared difference is used in place of the auto-correlation function due to the fact that the auto-correlation function handles changes in the signal amplitude over the course of a single analysis window poorly. If the signal increases in amplitude over an analysis window, the peaks of the auto-correlation function will grow as the lag grows rather than remaining constant. This causes the auto-correlation function to skip over the peak corresponding to the period in favor of larger peaks corresponding to larger lags. The squared difference function is immune to this problem, as the changes in amplitude in an analysis window affect all lag sizes equally.

The results of the squared difference function are then further refined by normalizing them with the cumulative mean in the following manner:

$$d'_t(\tau) = y = \begin{cases} 1 & \text{if } \tau = 0 \\ \frac{d_t(\tau)}{(1/\tau)\sum_{j=1}^{\tau} d_t(j)} & \text{otherwise} \end{cases}$$

The primary benefit of normalizing the results of the squared difference function is that it allows one to search for the first minimum that crosses a certain given threshold rather than the absolute minimum of all the lag values. It can happen that there is a dip in the signal (typically near a period that is at an octave of the period's frequency) that is deeper than the dip of the period. Selecting a threshold value and choosing the minimum of the first dip that crosses this threshold (or the absolute minimum if none is found) reduces this type of error.

Next, the results of the cumulative mean normalized difference function are then further refined by parabolically interpolating near the period to get a better estimate of the minimum. This noticeably improves the accuracy of the period estimates for high frequency signals. And finally, the signal is reexamined within a restricted range encompassing different phase offsets of the period to obtain the final estimate (Cheveigne, 2001).

a.) Accuracy of the algorithm

When compared to other pitch detection algorithms by Cheveigne using four speech databases, Yin was found to have the lowest error rate of all methods. 99% of its estimates were accurate within 20% of the ground truth value. 94% were accurate to 5%, and 60% were accurate to 1%.

The author further evaluated his own C++ implementation of Yin using a database of eighty-seven violin samples where only the steady-state portion of the signal was used. (The transient portion of the signal as explained in the section on violin acoustics does not have a clear pitch, and therefore, one should not expect any pitch tracker to accurately identify its pitch.) In the initial evaluation Yin correctly identified the pitch for 99.24% of the windows to within 1% of the ground truth. The errors that Yin made were exclusively octave errors. When the author more closely examined the recordings where Yin made detection errors, it

became apparent that in those recordings which were of the E4 note the violin's E5 string had not been damped and the E5 string was ringing sympathetically with the E4 causing the octave errors. The author rerecorded the E4 samples, and in a subsequent evaluation Yin correctly identified the pitch for 100% of the windows. This formal evaluation confirmed the impression that the author had made during the informal evaluation: for steady-state violin signals, the Yin algorithm does not make prediction errors.

One caveat, however, needs to be given for the above statement. The signal provided to Yin must be loud enough to rise above the noise floor. And as the violinist moves up the finger board of the violin shifting from the lower positions to the higher positions, the maximum possible amplitude of each note decreases as the string length decreases. This means that notes played at higher positions tend to be softer, and therefore, to get accurate pitch tracking results in these positions, one may need to use a compression – limiter which unfortunately comes with the drawback of decreasing the dynamic range of the violin.

b.) Performance Considerations

Autocorrelation algorithms including Yin are computationally expensive. The Yin algorithm as presented by Cheveigne requires n^2 operations where n is the window size. There are, however, a variety of methods to reduce the algorithms computational cost.

Two methods for doing so were suggested by Cheveigne in (Cheveigne, 2001). The first method involves using a recursive division of powers algorithm similar in nature to that which underlies the FFT. The second method involves using the FFT itself by implementing Yin as a spectral domain algorithm. A third method not mentioned by Cheveigne known as fast autocorrelation was implemented by the author (Middleton, 2003). In fast autocorrelation the results of the last autocorrelation pitch estimate are stored, and the algorithm initially computes only the

autocorrelation values in the immediate vicinity of the last value for the next evaluation. If one of these values crosses the threshold value, then it is used as the pitch value and the remaining autocorrelation values are not computed, which saves a considerable number of CPU cycles. But if none of the values in the immediate vicinity of the last estimate cross the threshold value, then all of the remaining values are computed. This method is possibly more efficient than those mentioned by Cheveigne, but it comes with the drawback that a reliable value for the aperiodicity of the signal cannot be computed, because in order to compute the aperiodicity, the autocorrelation values for all of the lags of the window are needed. Whether this drawback is important, however, depends on whether the aperiodicity value is needed by a particular application.

B. Spectral domain features

We would like to extract as much control information from the violin as possible. As violinists make choices about the bow bridge distance, bow pressure, bow tilt, and bow position in order to influence the quality of the tone produced, we would like to use as much of this information as possible to influence the quality of tone during the synthesis. As mentioned earlier in the section on violin acoustics, adjustments to bow bridge distance, bow pressure, bow tilt and bow position all affect the brightness of the tone; the sound of the violin becomes brighter as the bow pressure increases and / or as the bow bridge distance decreases. Therefore, we would like to extract the brightness of the bowed string and ideally represent it with a variable ranging from 0 – 1. But for this to be possible, we will need to develop a descriptor for the brightness of a bowed string that has a high correlation to the string's brightness and a small standard deviation.

Oftentimes, the brightness of a signal is modeled by calculating the spectral centroid of the FFT of the signal (Jehan, 2001). The spectral

centroid can be understood as the center of gravity of the STFT; if the STFT bins were to be split into two equally weighted halves, it is the point where they would be split. It is defined as follows:

$$c_j = \frac{\sum f_j a_j}{\sum a_j}$$

The spectral centroid given in this formulation does not, however, do a very good job, of modeling the changes in brightness of a bowed string that result from changes in the bow bridge distance and the bow pressure, because changes in the mean spectral centroid values show little correlation with changes in these bowing parameters. To prove this, the author created a database of sixty violin notes consisting of fives notes on each of the four strings being played three times with progressively decreasing bow bridge distances and increasing bow pressures and hence with progressively increasing brightness. He then calculated the correlation factor between each set of three recordings of a note and the spectral centroids of the STFTs of the three recordings. The total mean correlation for the entire database was found to be -0.3435. This implies that the spectral centroid tends to increase as the bow bridge distance decreases, which is in accordance with the literature on violin acoustics, but the correlation between the two variables is fairly weak. The spectral centroid also has a fairly large standard deviation even when the bowing parameters which have the greatest influence on brightness are held constant and brightness does not change perceptibly. This suggests that the source of the deviation lies elsewhere.

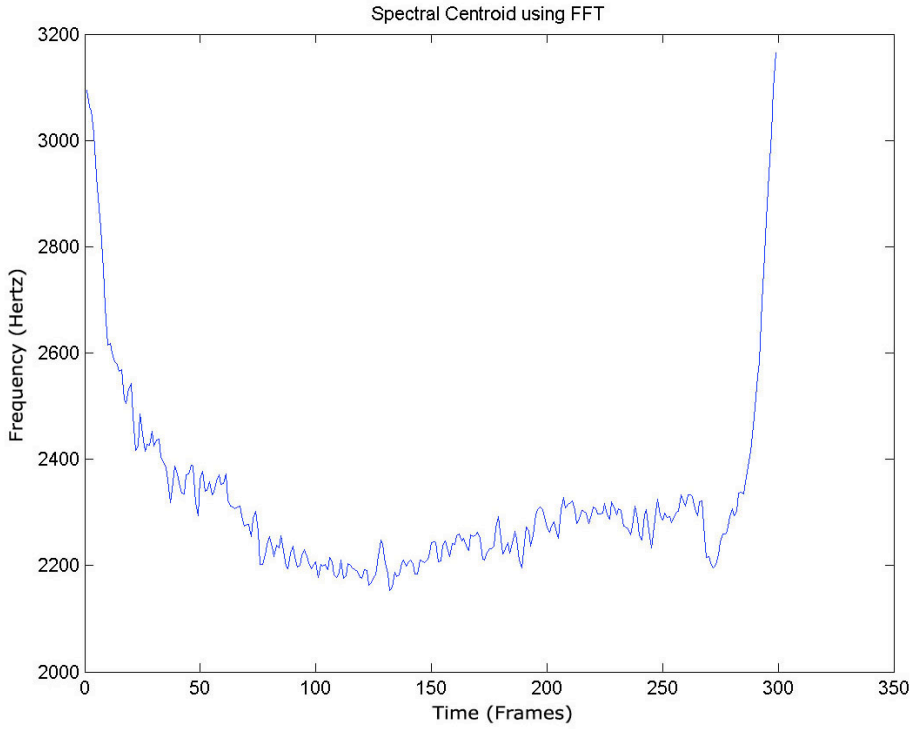


Figure 6 Spectral Centroid for a single C4 note played on a Zeta violin

Looking at the example spectral centroid in the figure above, the source of this instability should be fairly obvious. There is a sharp spike in the spectral centroid at the start of the note and at the end of the note. During these periods the bowed string is in a transient phase, and the motion of the string is highly aperiodic and noisy. As the spectral centroid of noise is much higher than the spectral centroid of a bowed string in Helmholtz motion[1], when the string enters a transient stage, the spectral centroid rises.

We would like to separate changes in brightness that occur due to changes in bow bridge distance and bow pressure from changes in brightness that occur due to changes in the noiseness of the signal. In order to do so, we need to limit the scope of the validity of the brightness descriptor to Helmholtz motion alone, i.e. during transient portions of the signal, the brightness descriptor need not be calculated. And during Helmholtz motion, we need to calculate the brightness of the signal based

[1] The spectral centroid of white noise is equal to half of the Nyquist frequency, which for a sampling rate of 44,100 would be 11,025 Hertz.

only on the changes in the sharpness of the Helmholtz corner as described in the preceding section on violin acoustics.

In order to follow changes in the sharpness of the Helmholtz corner in the time domain, we need to follow changes in those peaks in the frequency domain, which are multiples of the fundamental frequency.

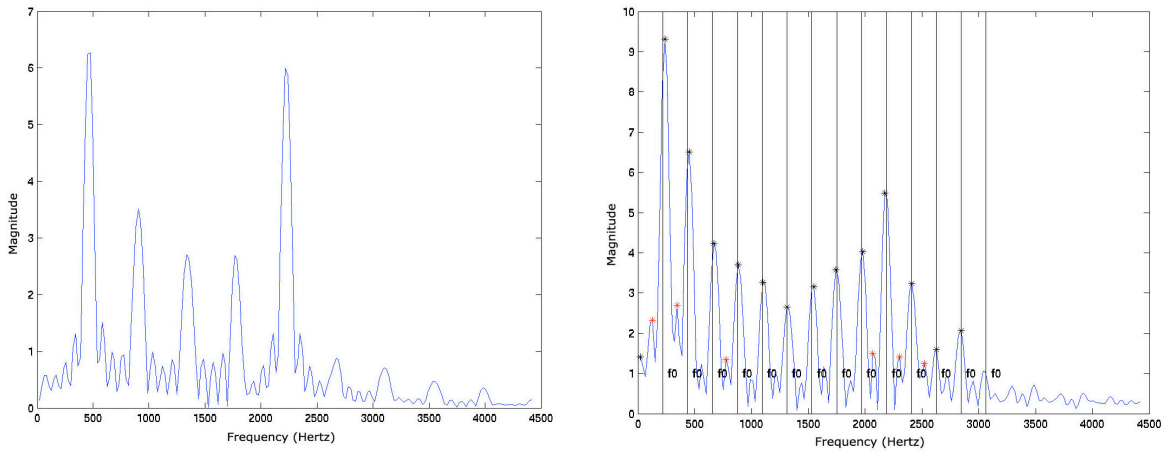


Figure 7 The spectrum of an A4 note played on a Zeta violin with an illustration of the peak selection algorithm used for calculating the centroid of the peaks. The vertical black lines in the second graph denote multiples of the fundamental. The peaks selected by the algorithm are topped with black asterisks, while the peaks ignored by the algorithm are topped by red asterisks.

As shown in the figure above, we select the largest peaks of those peaks which are close to a multiple of the fundamental frequency and calculate their spectral centroid. When tested using the same database used for the spectral centroid of the STFT, the correlation then climbs to -0.84, which is a considerable improvement over the previous correlation value, and the standard deviation falls significantly as well.

There is a difficulty, however, that comes with using the spectral centroid of the peaks as a metric for brightness. The number of peaks decreases as the fundamental frequency increases, and since there are less peaks to use in the calculation of the peak centroid, the peak centroid also decreases as the fundamental frequency increases (after being normalized by dividing its value by that of the fundamental frequency). It might be possible to define a function to normalize the

peak centroid using either the fundamental frequency of the signal or the number of peaks used to calculate the centroid, but one could also perfectly well use another descriptor which would be unaffected by changes in the number of peaks used to calculate its value.

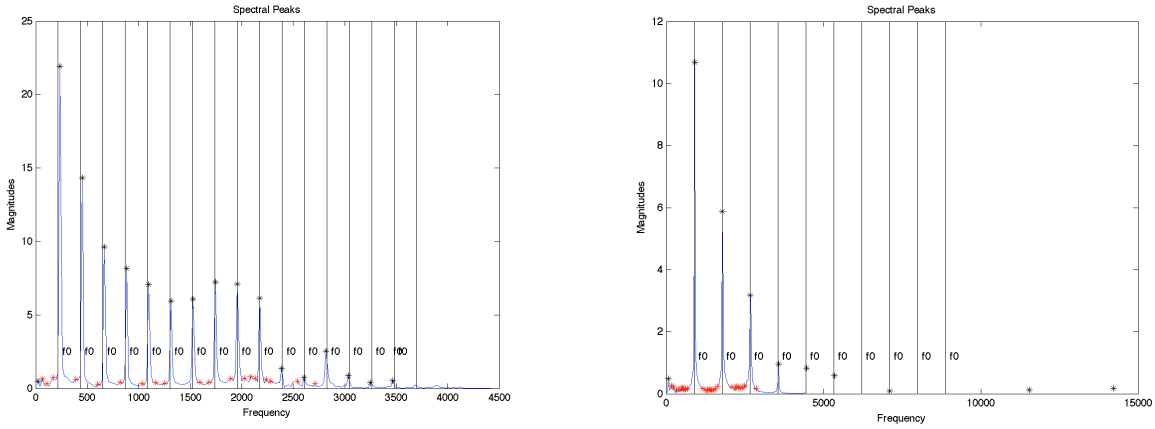


Figure 8 The spectrum of an A3 and an A5 note played on a Zeta violin. As can be seen, the A5 has significantly fewer peaks than the A3.

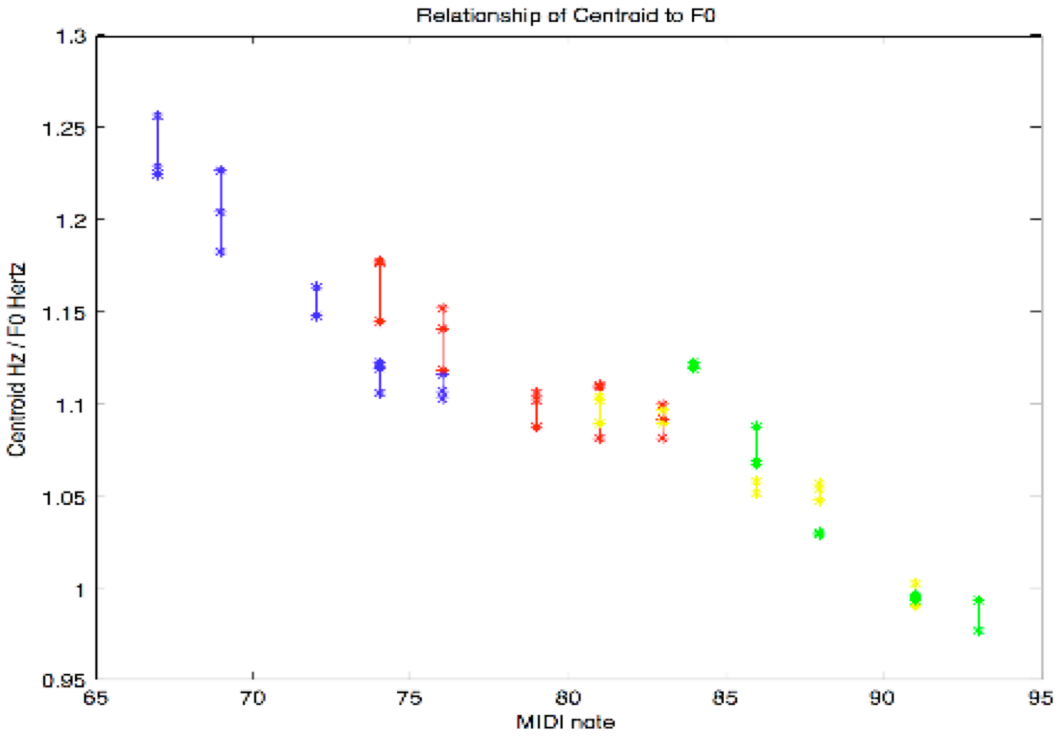


Figure 9 The mean peak centroid values of each entire note. Each note was played with three brightness values, and the three notes with their three brightness values are shown connected by a line. The notes were color coded by string. The blue asterisks represent notes played on the violin's G string – the lowest string. The red asterisks represent notes played on the violin's D string, the yellow asterisks notes on the A string, and the green asterisks notes on the E string – the highest string. As can be seen from the graph, the peak centroid falls as the fundamental frequency rises.

Let us consider for a moment what form another descriptor for tracking changes in brightness might take. As the brightness increases, the magnitudes of the peaks to the right of the fundamental should increase and possibly the total number of peaks might increase as well. It stands to reason then that if one were to draw a line through the peaks then the slope of this line should decrease as the brightness increases. The descriptor just described is similar in nature to the descriptor normally referred as the spectral slope (Alastair et al, 2004). However, the spectral slope is usually calculated as the difference in energy between the lower and upper halves of the spectrum. To avoid confusion, the author will instead refer to slope of the line that passes through the selected peaks as given by the least squares method (Weisstein, 2007) as

the spectral peak slope.

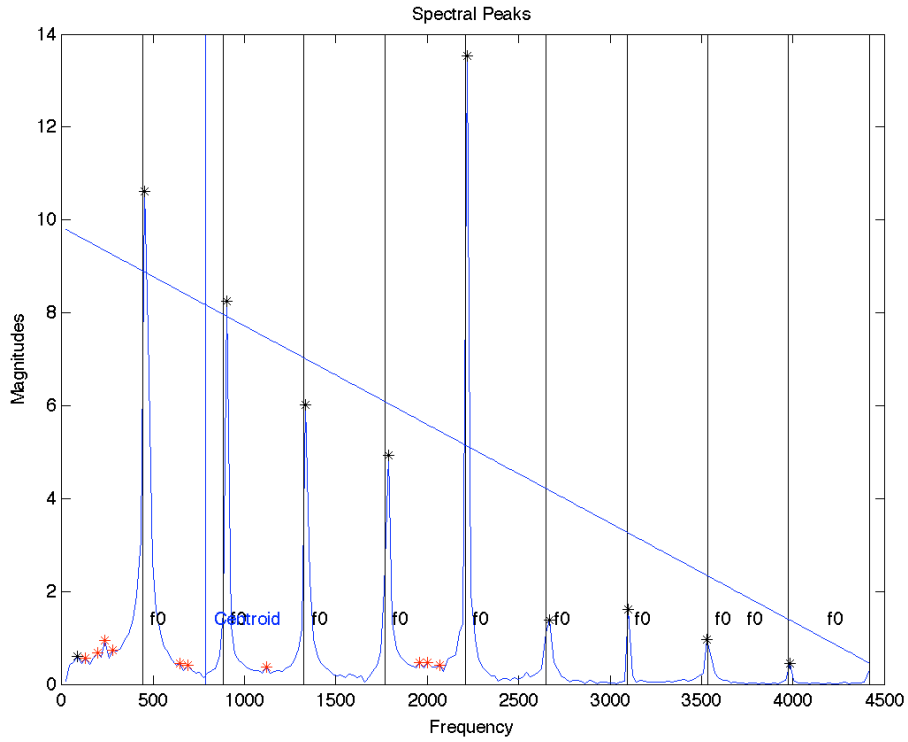


Figure 10 The spectral peak slope of an A4 note played on a Zeta violin with a small bow bridge distance and high bow pressure.

The spectral peak slope does indeed prove to be more effective at tracking the brightness than the spectral peak centroid. It yielded a correlation value of -0.9613 which is a considerable improvement over the correlation value of the spectral peak centroid and which is very close to the ideal value of -1. And after scaling the slope to range from 0 – 1, it had a standard deviation of 0.0387. The author attempted to further reduce the standard deviation by applying sinusoidal modeling techniques such as peak continuation across frames, but applying these techniques only marginally improving the standard deviation while significantly reducing the correlation. The author also tried different schemes for weighting the peaks when calculating the slope, but he found that weighting the peaks equally gave better results. The author found that the following peak selection / slope calculation algorithm gave the best

results: (1) generate the maximum number of peaks to be selected as a function of the fundamental frequency so that as the fundamental frequency increases the maximum number of peaks falls (2) for every multiple of the f0 select the largest peak which is within a certain threshold distance from the multiple's location where the threshold is defined to be 20% of the fundamental frequencies bin position (3) calculate the spectral peak slope using the interpolated magnitudes and locations of the selected peaks where every peak is weighted equally.

The author validated these results by testing how well the spectral peak slope worked as a classifier of samples recorded with differing bow bridge distances. As brightness varies with bow bridge distance, we can use different bow bridge distances as a proxy for the ground truth of different brightness values. The author recorded 24 notes where the bow was near to the bridge and far from the bridge. (Middle bridge distances were not used, because notes played at this position can be identically bright to notes played at other bow bridge positions if changes in the bow pressure offset changes in the bow bridge distance.) The notes were classified with 88.89% accuracy if the string which the note was played on was given. Otherwise, they were classified with a 77.78% accuracy.

Information about which string a note is played on proved to be important, because different strings are stretched to different pressures, and the same note played on different strings creates a different number of peaks with different magnitudes and hence different spectral peak slopes. As the mean values for the spectral peak slopes changes from one string to the next, these values are best scaled to the range 0 – 1 using functions which are specific to the string. Although a mapping function was also derived for all strings, it obviously did not work as well. In end effect this means that violinists playing on violins with one pickup per string will have better results than violinists playing with only a single pickup, but having one pickup per string already gave better results in the sense that it allows violinists to play polyphonically.

C. Bow Changes

Oftentimes before a violinist plays a new piece, they look at the score first in order to plan their bow movements. Notes in the score that require emphasis and therefore greater bow pressure are best played close to the frog, i.e. the base of the bow, where one can summon the greatest bow force, and notes that call for a soft onset are best played at the tip of the bow where the chances of losing steady contact with the string due to an unintentional movement of the hand are lessened. The choice of points in the score at which to change the bow direction is also important as notes played on the same bow tend to be heard as phrases. A violinist may prefer for example to play all the notes of a melody leading to a melodic apex on one bow while playing the subsequent descending notes on another bow. This helps create the impression that the violinist is playing towards the inflection points in the score which lends the piece a flowing quality. Poorly planned bow changes on the other hand can cause a piece to sound choppy and unmusical.

In terms of acoustics, the likely reason why violinists plan their bow changes so carefully is that the string undergoes a lengthier transient phase when the bow changes. A bow change is the only time when a string must be brought into Helmholtz motion from still stand by a bow accelerating from still stand as well. In those cases where a note is played on the same bow, the bow is in any case already moving and possibly the string is already moving as well (if the previous note was played on the same string).

The bow direction and changes in the bow direction should be treated as significant later on when we synthesize a new signal using the controls extracted from the violin signal. As a bow change usually corresponds to a lengthy transient, we should correspondingly play a full attack for a synthesized note whenever there is a bow change. For this reason we would like to be able to detect the bow direction and the

changes in bow direction.

Given only an airborne signal recorded by a microphone, detecting the bow direction might be rather difficult, but this task becomes much easier if one uses a signal recorded by a pick-up in direct contact with the string, because a pick-up is actually pulled with the string by the bow during an up or a down bow and hence the signal is centered either above or below zero depending on the bow direction.

Is it unreasonable to expect that the signal will be recorded by a pickup? Every electric violin that the author is aware of uses pickups. And as only an electric violin will produce a signal soft enough as to not drown out the signal produced by the synthesizer, one can safely say that an electric violin is the optimal type of violin for an audio-driven synthesis application. Although it might be preferable to develop a method that would work independently of the manner in which the signal was recorded, as the reader will soon see, it is unlikely that such a method would work as well as a pickup specific method.

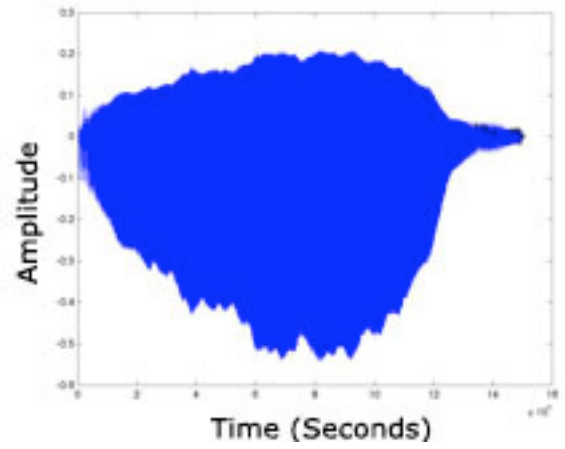


Figure 11 Down bow.

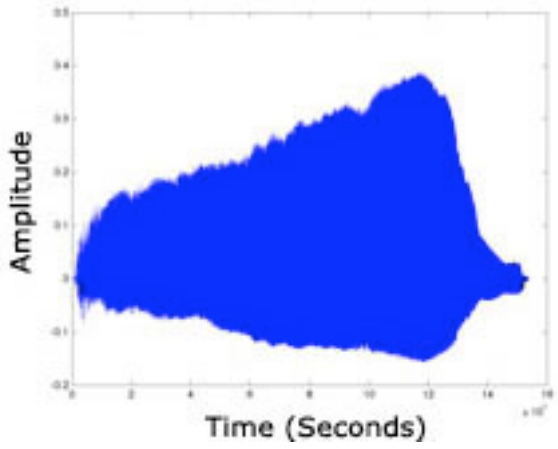


Figure 12 Up bow.

A number of different measures were considered to extract the bow direction from a window of the signal, and these measures were subsequently tested on eight Zeta violin notes. The first measure was to compute the mean value of a window of the signal and extract the bow

direction from whether the mean was positive or negative. The second measure was to find the maximum and the minimum of a window and to extract the bow direction based on whether the maximum exceeded the absolute value of the minimum. The third measure was to find the average of the five largest maximums and the average of the five smallest minimums of a window and to extract the bow direction based on whether the mean maximum exceeded the absolute value of the mean minimum.

The first measure simply did not work as there was little correlation between the mean of a window and the direction of the bow. The second measure accurately predicted the down bows for 96.12% of the windows and the up bows for 98.05% of the windows. The third measure had more or less the same accuracy. It correctly predicted the down bows for 96.09% of the windows and the up bows for 98.08% of the windows. As the second measure is computationally less expensive than the third measure, it is therefore to be preferred.

These numbers suggest that both the second and third measures identify the bow direction fairly accurately. But as the errors that they made were exclusively during the transient portion of the signal, they could be even further improved by suppressing these values during the transient portion of the signal. (If one looks closely at the figures above, small black asterisks in the signal identify the points where the bow direction was incorrectly identified.)

D. Transient detection

Accurate identification of transients in the signal is essential for any audio driven synthesis application, because when transients occur, the pitch values delivered by the pitch detection algorithm are inaccurate, oftentimes substantially so. If they are not filtered out, one may hear a flurry of wild pitch values with nearly every note change. Transients

must, therefore, be identified as such so that they can be suppressed or dealt with in some other way.

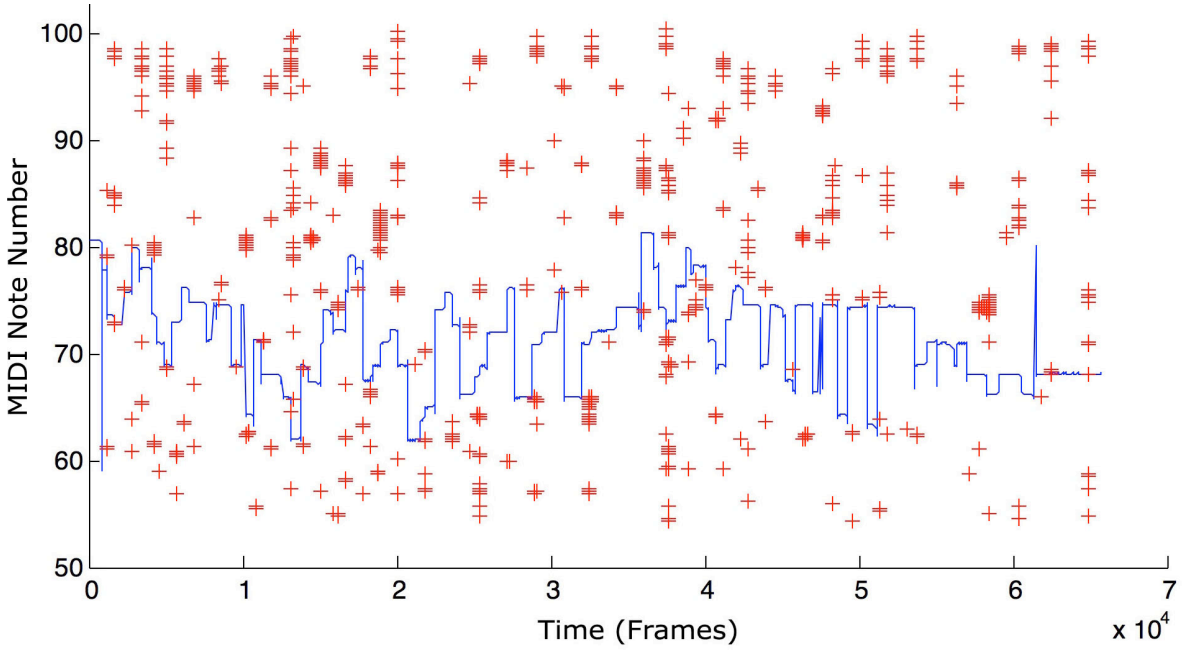


Figure 13 Stationaries and transients in a violin performance. The red plus signs represent bad pitch estimates from Yin during a transient.

The author evaluated four different measures for identifying transients. First, as previously mentioned, the yin algorithm and other auto-correlation algorithms provide a measure of the aperiodicity of the signal. When transients occur, there is typically a spike in the aperiodicity of the signal, and this value has been used by at least one acoustician as the basis of a transient identification algorithm developed for bowed strings (Woodhouse, 2003). Second, when transients occur, Yin's pitch estimates oftentimes deviate substantially from the previous stable pitch value, and therefore, rather than trying to identify transients by looking at the signal itself, one could instead look at Yin's output pitch values for pitch changes that cross a threshold level. Third, the author noticed while developing the brightness descriptor that when transients occur, the spectral peak centroid oftentimes deviates significantly from its steady state value for a particular pitch, and therefore, checking to see whether

the spectral centroid falls outside of a normal range of values is a further potentially useful measure. Fourth, the author developed a measure of the distance of the current pitch value from the most recent pitch values. The measure works by maintaining a running histogram of the last half second of pitch values so that the distance of the current value from the recent values can be calculated as the cost of moving all the weights in the other histogram bins into the bin of the current pitch. This cost is then subsequently normalized by dividing it by the largest possible cost. This distance measurement borrows conceptually from the earth mover's distance algorithm (Rubner, et al, 1998).

In order to test the efficacy of these four measures, the author played, recorded, and transcribed three solo violin pieces. Oftentimes, in the transition between notes, it was not possible to say exactly when one pitch ended and another began which means that the ground truth could not be fully determined by transcription. But the ground truth could be approximated during the note transitions by declaring any pitch values between the first and the second pitches in a transition to be good estimates, and any pitches outside that range to be poor estimates.

In order to evaluate these four measures, the author used each of the four measures to classify each frame of the signal as either stationary or transient, and each classification was compared to the ground truth to assess its accuracy. At the end of the evaluation, rather than calculating one number representing the total percent correct, the author calculated four numbers: the percentage of correct stationary classifications, the percentage of incorrect stationary classifications, the percentage of correct transient classifications, and the percentage of incorrect transient classifications. The reason for dividing the results into these four categories is that although an incorrect stationary classification and an incorrect transient classification are both errors, incorrectly labeling a transient as a stationary is a far worse error than incorrectly labeling a stationary as a transient. When we incorrectly label a transient as a stationary, this leads us to synthesize a new note using a bad pitch

estimate. When we incorrectly label a stationary as a transient on the other hand, the value will most likely be suppressed which means that we will continue to sustain the value of the current note at its current pitch and amplitude until the next stationary. In end effect we introduce additional latency to the onset of new notes, but this is a far less severe error than outputting a bad note. As the reader will see for each of the measures evaluated, there is a trade-off between these two types of errors. By adjusting the threshold values, one can decrease one type of error but only at the cost of increasing the other type of error.

In the following tables the data is given for how successfully each measure works to classify frames as either stationary or transient. Different threshold values for each measure were used to classify frames as stationary or transient, and the percentage of correct classifications that resulted is given for each measure. For each measure two tables are given. The first gives the percentages of stationery frames or transient frames identified with respect to the total number of stationeries or transients frames, and the second gives the percentages of stationery or transient frames identified with respect to the total number of frames. Before applying any measures, the recordings evaluated consisted of 94.92% stationary frames and 5.08% transient frames[2].

1.) Transient detection using aperiodicity

[2] These numbers overstate, however, to an extent the number of transient frames as in many cases transient frames occur during moments of near silence. As the amplitude level is so low for these frames, it is somewhat irrelevant if the pitch is correctly or incorrectly estimated as they will not be heard anyway. These nearly silent frames account for 10 – 30% of the transients.

Aperiodicity Threshold	Correct Stationary Classification	Incorrect Stationary Classification	Correct Transient Classification	Incorrect Transient Classification
0.0500	91.91%	8.08%	59.80%	40.19%
0.0700	93.60%	6.39%	52.02%	47.97%
0.0900	94.78%	5.21%	46.31%	53.68%
0.2000	97.73%	2.26%	28.66%	71.33%
0.3000	98.96%	1.03%	20.77%	79.22%
0.4000	99.48%	0.51%	14.19%	85.80%
0.5000	99.76%	0.23%	10.28%	89.71%
0.6000	99.87%	0.12%	7.16%	92.83%
0.7000	99.91%	0.08%	4.91%	95.08%
0.8000	99.93%	0.06%	3.68%	96.31%
0.9000	99.96%	0.03%	2.50%	97.49%
1.0000	99.97%	0.02%	1.74%	98.25%

Table 1 *In this table as the allowed amount of aperiodicity increases, the percentage of correctly identified stationary frames increases while the percentage of correctly identified transient frames decreases. The percentages indicate the percentages of correctly identified stationery frames with respect to the total number of stationary frames and the percentage of correctly identified transient frames with respect to the total number of transient frames.*

Aperiodicity Threshold	Correct Stationary Classification	Incorrect Stationary Classification	Correct Transient Classification	Incorrect Transient Classification
0.0500	86.88%	7.64%	3.27%	2.19%
0.0700	88.48%	6.04%	2.84%	2.62%
0.0900	89.60%	4.92%	2.53%	2.93%
0.2000	92.39%	2.13%	1.56%	3.90%
0.3000	93.54%	0.97%	1.13%	4.33%
0.4000	94.04%	0.48%	0.77%	4.69%
0.5000	94.30%	0.22%	0.56%	4.90%
0.6000	94.40%	0.12%	0.39%	5.07%
0.7000	94.45%	0.07%	0.26%	5.20%
0.8000	94.47%	0.05%	0.20%	5.26%
0.9000	94.49%	0.03%	0.13%	5.33%
1.0000	94.50%	0.02%	0.09%	5.37%

Table 2 *In this table the percentages of correctly and incorrectly classified frames is given with respect to the total number of frames.*

2.) Transient detection using pitch changes

Pitch Change Threshold	Correct Stationary Classification	Incorrect Stationary Classification	Correct Transient Classification	Incorrect Transient Classification
1.0006	85.73%	14.26%	63.07%	36.92%
1.0012	92.61%	7.38%	51.55%	48.44%
1.0017	94.97%	5.02%	44.71%	55.28%
1.0023	96.29%	3.70%	40.19%	59.80%
1.0029	97.05%	2.94%	36.69%	63.30%
1.0035	97.60%	2.39%	33.92%	66.07%
1.0041	97.95%	2.04%	31.55%	68.44%
1.0046	98.24%	1.75%	29.54%	70.45%
1.0052	98.45%	1.54%	28.07%	71.92%
1.0058	98.61%	1.38%	26.55%	73.44%
1.0116	99.35%	0.64%	19.37%	80.62%
1.0175	99.48%	0.51%	16.07%	83.92%
1.0234	99.55%	0.44%	14.51%	85.48%
1.0293	99.58%	0.41%	13.55%	86.44%
1.0353	99.61%	0.38%	13.27%	86.72%
1.0413	99.62%	0.37%	12.93%	87.06%
1.0473	99.62%	0.37%	12.42%	87.57%
1.0534	99.62%	0.37%	11.97%	88.02%
1.0595	99.62%	0.37%	11.72%	88.27%

Table 3 *In this table as the allowed size of the change in pitch between frames decreases, the percentage of correctly identified stationary frames decreases while the percentage of correctly identified transient frames increases. The percentages indicate the percentages of correctly identified stationery frames with respect to the total number of stationary frames and the percentage of correctly identified transient frames with respect to the total number of transient frames. The pitch change threshold is the ratio of the current pitch to the previous pitch, and the value 1.05946 represents one semitone.*

Pitch Change Threshold	Correct Stationary Classification	Incorrect Stationary Classification	Correct Transient Classification	Incorrect Transient Classification
1.0006	81.06%	13.49%	3.43%	2.01%
1.0012	87.56%	6.98%	2.80%	2.63%
1.0017	89.80%	4.75%	2.43%	3.01%
1.0023	91.05%	3.50%	2.18%	3.25%
1.0029	91.77%	2.78%	1.99%	3.44%
1.0035	92.28%	2.26%	1.84%	3.59%
1.0041	92.62%	1.92%	1.71%	3.72%
1.0046	92.89%	1.66%	1.60%	3.83%
1.0052	93.08%	1.46%	1.52%	3.91%
1.0058	93.24%	1.31%	1.44%	4.00%
1.0116	93.93%	0.61%	1.05%	4.39%
1.0175	94.06%	0.48%	0.87%	4.57%
1.0234	94.13%	0.41%	0.79%	4.65%
1.0293	94.16%	0.38%	0.73%	4.70%
1.0353	94.18%	0.36%	0.72%	4.72%
1.0413	94.19%	0.35%	0.70%	4.74%
1.0473	94.19%	0.35%	0.67%	4.77%
1.0534	94.19%	0.35%	0.65%	4.79%
1.0595	94.20%	0.35%	0.63%	4.80%

Table 4 *In this table the percentages of correctly classified frames is given with respect to the total number of frames.*

3.) Transient detection using the spectral peak centroid

Distance from mean value	Correct Stationary Classification	Incorrect Stationary Classification	Correct Transient Classification	Incorrect Transient Classification
0.10	96.44%	3.55%	47.18%	52.81%
0.15	99.19%	0.80%	31.42%	68.57%
0.20	99.57%	0.42%	12.73%	87.26%
0.25	99.68%	0.31%	8.71%	91.28%
0.30	99.80%	0.19%	4.74%	95.25%
0.35	99.87%	0.12%	2.50%	97.49%
0.40	99.92%	0.07%	1.51%	98.48%

Table 5 *In this table as the allowed size of the distance of the spectral peak centroid from the mean spectral peak centroid increases, the percentage of correctly identified stationary frames increases while the percentage of correctly identified transient frames decreases. The percentages indicate the percentages of correctly identified stationery frames with respect to the total number of stationary frames and the percentage of correctly identified transient frames with respect to the total number of transient frames.*

Distance from mean value	Correct Stationary Classification	Incorrect Stationary Classification	Correct Transient Classification	Incorrect Transient Classification
0.10	91.16%	3.35%	2.58%	2.88%
0.15	93.76%	0.76%	1.71%	3.75%
0.20	94.12%	0.40%	0.69%	4.77%
0.25	94.23%	0.29%	0.47%	4.99%
0.30	94.34%	0.18%	0.25%	5.21%
0.35	94.41%	0.11%	0.13%	5.33%
0.40	94.45%	0.07%	0.08%	5.38%

Table 6 *In this table the percentages of correctly classified frames is given with respect to the total number of frames.*

4. Transient detection using a running histogram

Distance from recent values	Correct Stationary Classification	Incorrect Stationary Classification	Correct Transient Classification	Incorrect Transient Classification
0.07	73.52%	26.47%	72.66%	27.33%
0.08	83.17%	16.82%	68.79%	31.20%
0.09	88.47%	11.52%	66.29%	33.70%
0.10	92.17%	7.82%	61.60%	38.39%
0.11	95.69%	4.30%	55.71%	44.28%
0.12	97.89%	2.10%	52.67%	47.32%
0.13	98.77%	1.22%	50.51%	49.48%
0.14	99.01%	0.98%	44.82%	55.17%
0.15	99.24%	0.75%	43.23%	56.76%
0.16	99.47%	0.52%	36.29%	63.70%
0.17	99.71%	0.28%	31.42%	68.57%
0.18	99.96%	0.03%	29.26%	70.73%
0.19	99.98%	0.01%	27.61%	72.38%

Table 7 *In this table as the allowed distance (measured as transformational cost) of the current pitch value from the most recent values increases, the percentage of correctly identified stationary frames increases while the percentage of correctly identified transient frames decreases. The percentages indicate the percentages of correctly identified stationery frames with respect to the total number of stationary frames and the percentage of correctly identified transient frames with respect to the total number of transient frames.*

Distance from recent values	Correct Stationary Classification	Incorrect Stationary Classification	Correct Transient Classification	Incorrect Transient Classification
0.07	69.79%	25.12%	3.68%	1.38%
0.08	78.95%	15.97%	3.49%	1.58%
0.09	83.97%	10.94%	3.36%	1.71%
0.10	87.49%	7.42%	3.12%	1.94%
0.11	90.83%	4.08%	2.82%	2.24%
0.12	92.92%	1.99%	2.67%	2.40%
0.13	93.75%	1.16%	2.56%	2.51%
0.14	93.98%	0.93%	2.27%	2.80%
0.15	94.20%	0.71%	2.19%	2.88%
0.16	94.42%	0.49%	1.84%	3.23%
0.17	94.64%	0.27%	1.59%	3.48%
0.18	94.89%	0.03%	1.48%	3.59%
0.19	94.90%	0.01%	1.40%	3.67%

Table 8 *In this table the percentages of correctly classified frames is given with respect to the total number of frames.*

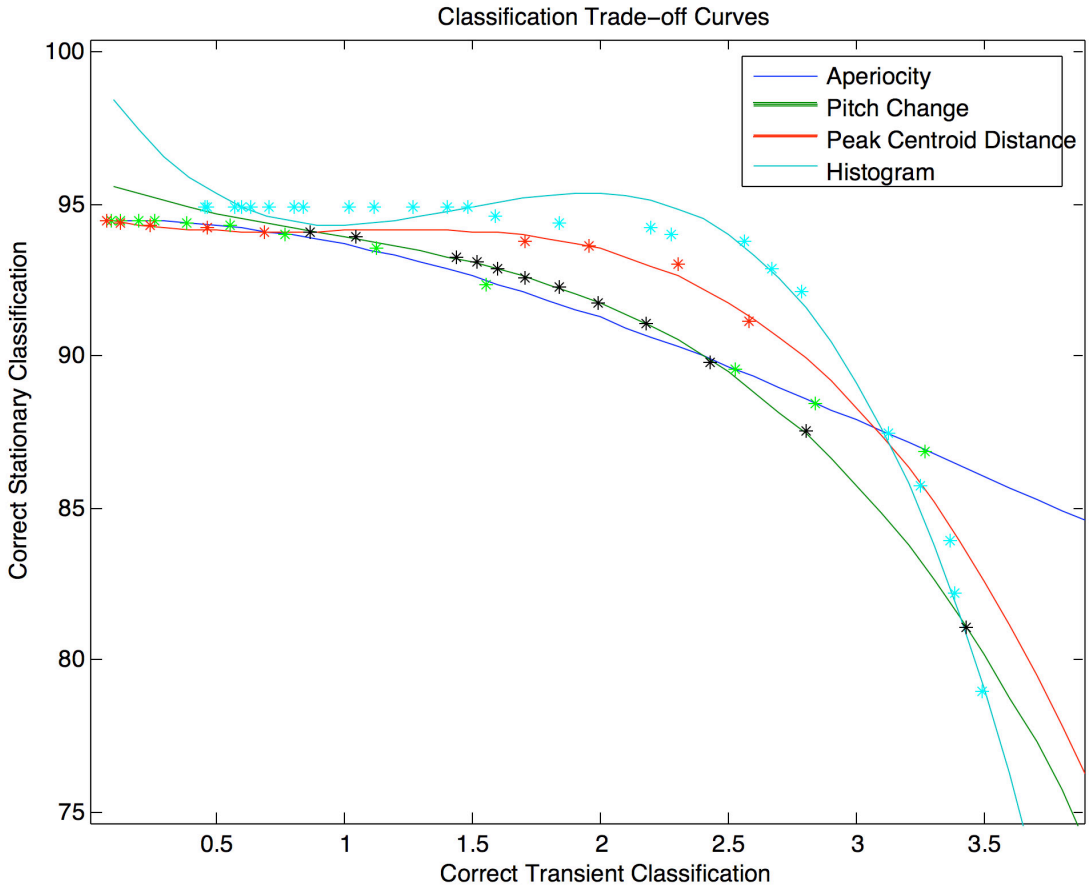


Figure 14 Curves showing the tradeoff for each measure between correct stationary and transient classification. Curves closer to the upper right hand corner present the best trade-off between correct stationary and transient classification. So, as can be seen, the histogram gives the best results while the aperiodicity and the pitch change give the worst results.

As none of these features proved to be sufficient to detect all of the transients on their own, we would like to combine these different features in the hopes of improving the overall accuracy of the transient classification. After having tried various classification algorithms in the Weka machine learning environment (Witten and Frank, 2005), the author chose to implement a naïve Bayesian classifier. Although the decision tree based classifiers in Weka gave slightly better results for this problem, implementation considerations lead the author to prefer the naïve Bayesian classifier.

Naïve Bayesian classifiers attempt to find the probability of a particular class given a set of features, $p(\text{Class} \mid F_1 \ldots F_n)$. In our particular case, the classes are whether a particular frame is a transient

or stationary, and the features are discretized values for the aperiodicity, the distance of the spectral peak centroid from the mean, and the distance of the current pitch value from recent values. (The pitch change was not included, because the running histogram did a better job of measuring essentially the same phenomena.) We can use Bayes' theorem to derive p(Class | F_1 … F_n) as:

$$p(C|F_1, ..., F_n) = \frac{p(C)\, p(F_1, ..., F_n|C)}{p(F_1, ..., F_n)}$$

And making the assumption from which naïve Bayes classification derives its name, we assume that F_1 … F_n are conditionally independent so that we can restate the above equation as:

$$p(C|F_1, ..., F_n) = \frac{p(C) \prod_{i=1}^{n} p(F_i|C)}{\prod_{i=1}^{n} p(F_i)}$$

We can then plug in the values of p(C), p(F_1 | Class), p(F_2 | Class), p(F_3 | Class), p(F_1), p(F_2), and p(F_3) to derive the probability of a particular frame being a stationary or a transient. Given these probabilities, rather than simply declaring the class with the greatest probability to be the winner, we use a user provided scaling value to determine by what percent the probability of a frame being a stationary must exceed the probability of a frame being a transient in order for that frame to be declared a stationary. The scaling factor can be used to select an appropriate trade-off between incorrectly identified stationeries and incorrectly identified transients as the percentages produced by the naïve Bayes' classifier exhibit the same trade-off between incorrectly identified stationeries and incorrectly identified transients as do its constituent features.

In order to compare the performance of the naïve Bayes' classifier with that of the individual measures, the author plotted the trade-off

curve between correct stationary classifications and correct transient classifications for all four measures as well as the naïve Bayes' classifier. One can judge how well the classifier works by looking at whether it presents a more favorable trade-off curve than its individual features.

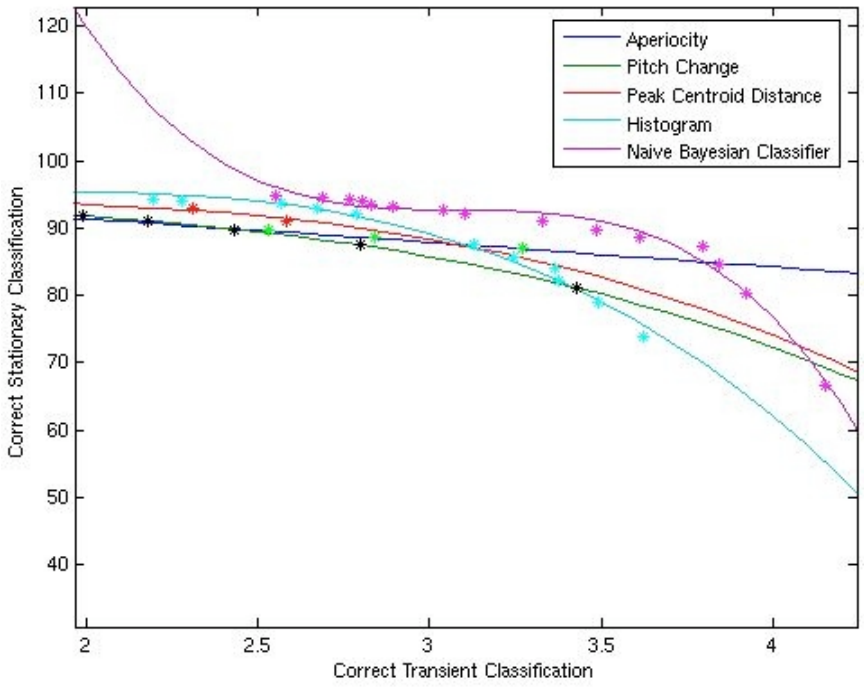


Figure 15 Trade-off curves between stationary and transient classification. Curves closer to the upper right hand corner present a better trade-off between errors in transient and stationary identification. As can be seen, the classifier performed significantly better than any of its constituent features.

The naïve Bayesian classifier did indeed perform better than its constituent features. It was able to correctly identify transients and stationeries with a significantly higher success rate than its constituent features. The overall accuracy of this classifier peaked at 97.4% while the overall accuracy of the most successful of its constituent features – the running histogram – peaked at 96.4%. Its overall error rate of 2.6% might still sound unacceptably high, but an examination of plots of the errors shows that the classifier manages to catch all of the egregious transients and what remains are primarily transients which are very close

to the actual pitch and mislabeled stationeries.

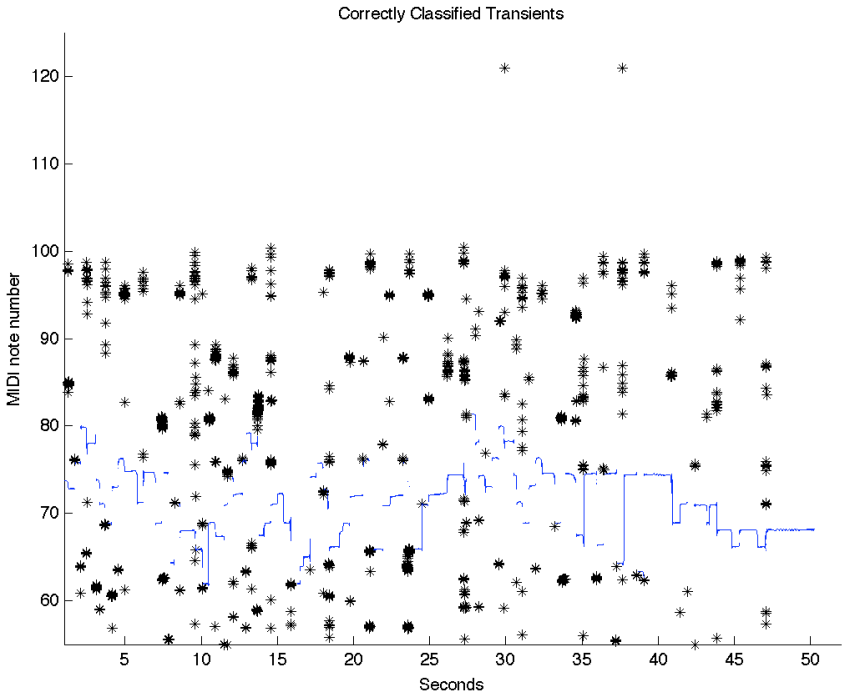


Figure 16 Depiction of correctly identified stationaries (in blues) and correctly identified transients (in black). The plot is for the point on the tradeoff curve where the overall accuracy is 95.2%.

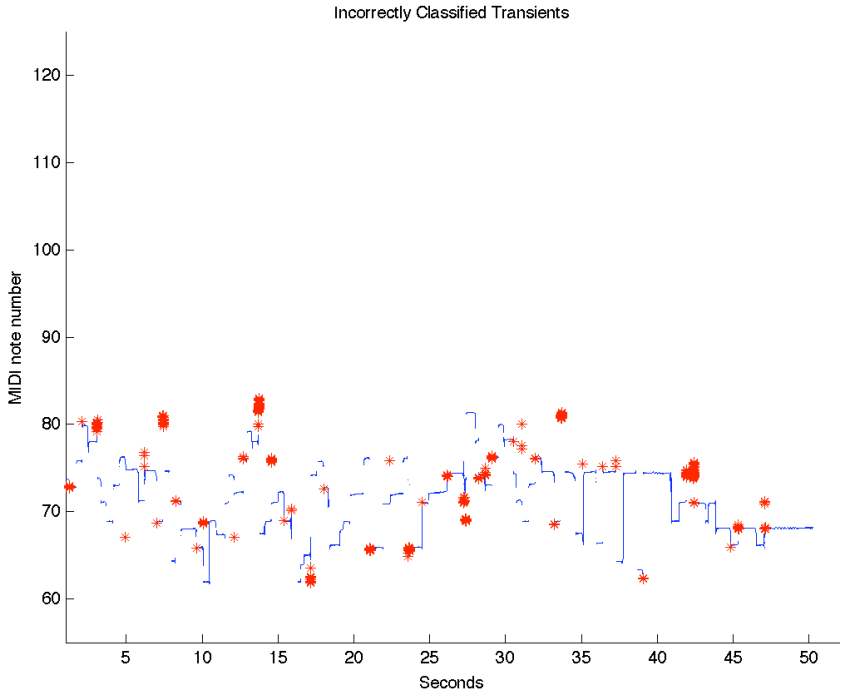


Figure 17 Depiction of incorrectly identified transients (in red) versus stationaries.

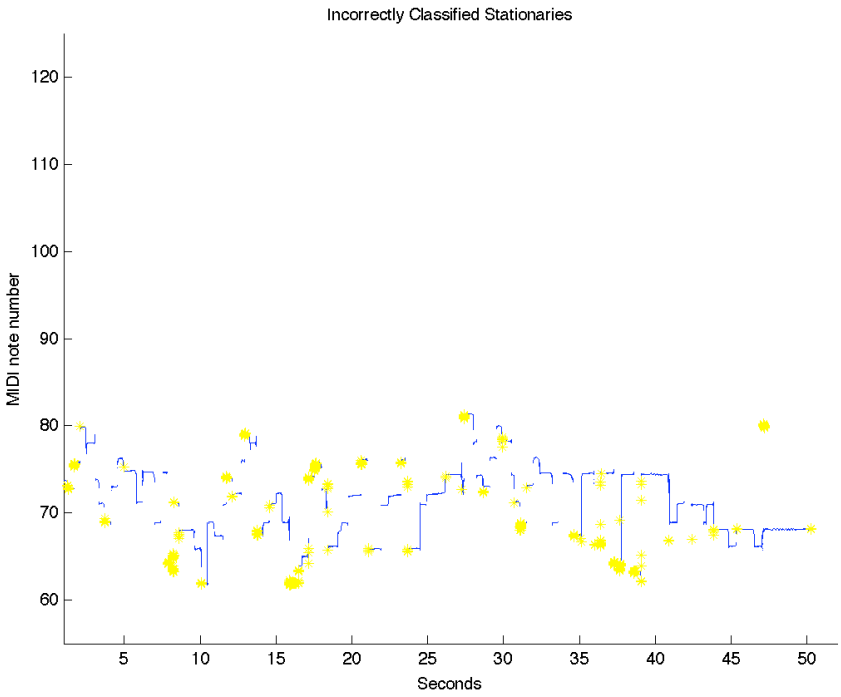


Figure 18 Depiction of incorrectly identified stationaries (in yellow) versus correctly identified stationaries.

VI. Spectral Modeling Synthesis Application

As previously stated in the introduction, one of the goals of this project was to create a real-time spectral modeling synthesis application, which works with samples provided by the musician. Before describing how we approached this goal, we will give a brief introduction to spectral modeling, and then, we will proceed to discuss the specifics of the application developed.

A. Background

A mathematical model is a representation of the essential aspects of an existing system (or a system to be constructed), which presents knowledge of that system in usable form.

Eykhoff (1974)

A spectral model of a signal is a frequency domain representation of a signal that enables us to perform a variety of transformations on a signal that might be achieved only with great difficulty or computational cost in the time domain. Spectral modeling encompasses the following three basic representations of a signal: the STFT (short time Fourier transform), sinusoidal models, and sinusoidal plus residual models. (There are many other representations in addition to these, but they can be seen as extensions to or variations of these three basic representations.) The models differ in how much flexibility in transformation they offer as well as in which sorts of signals they are able to effectively model. From these three representations, further high-level attributes of a signal can be extracted as well in order to aid certain transformations.

1. STFT

The discrete Fourier transform decomposes an audio signal into a continuous spectrum of its frequency components, or put differently, it models a signal as a sum of sine waves. The inverse transform synthesizes an audio signal from its spectrum of frequency components. Due to the fact that the Fourier transform does not give one any information about the temporal locations of frequency components, typically only a small window of the audio signal – a window just large enough to give adequate frequency resolution – is analyzed in order to localize the frequency components. Hence comes the name short time Fourier transform.

The STFT was first used for musical applications in the digital phase vocoder (Moorer, 1978) which is essentially a digital implementation of a fixed frequency filter bank. The phase vocoder is able to successfully analyze harmonic sounds with stable partials well enough to allow a variety of transformations of the signal without producing artifacts, but its analysis is less successful for inharmonic sounds or sounds with time varying frequency components. This is due to the fact that partials may wander between the bands of the filter bank or fall between the boundaries of two neighboring banks.

2. Sinusoidal Models

Sinusoidal modeling builds upon the STFT by identifying sinusoidal components in the frequency analysis, pinpointing their frequencies via interpolation, and tracking them across successive analysis windows. This allows for a better analysis of inharmonic sounds and sounds with time varying frequency components. However, it does not provide a very useful representation of noisy signals, as noise would basically have to be represented as a sinusoid at every frequency up to the Nyquist frequency. See (McAulay and Quatieri, 1986; Smith and Serra 1987) for further information on sinusoidal modeling.

3. Sinusoidal plus Residual Modeling

Sinusoidal plus residual modeling builds upon the sinusoidal model by performing the previous steps in sinusoidal modeling to obtain the sinusoidal components, which are then subtracted from the spectrum to yield a residual component, i.e. noise. The residual component can then be modeled by calculating its spectral envelope, which can later be used to re-synthesize the residual from white noise. Please see (Serra, 1997) for further information.

The following diagram gives a high level representation of the steps involved in a sinusoidal plus residual modeling analysis, transformation, and resynthesis process. Many different implementations are possible; the diagram depicts the implementation in the CLAM C++ library for audio and music (Amatriain, et al, 2006).

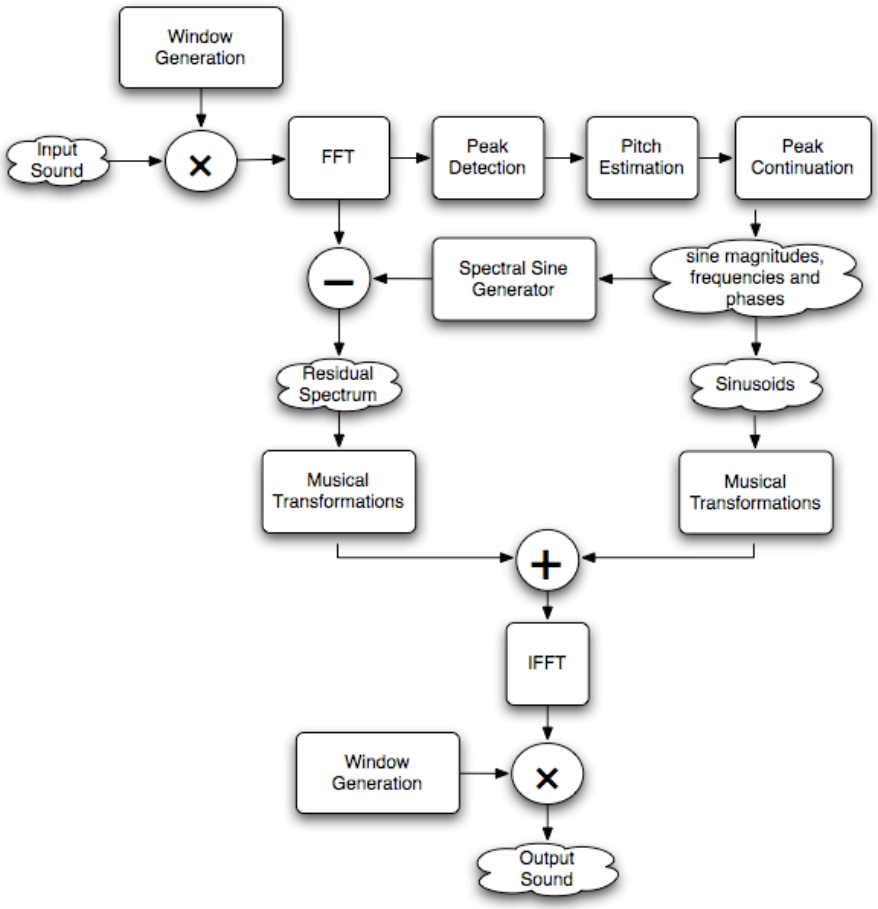


Figure 19 Diagram of the sinusoidal plus residual modeling process in CLAM

B. A Spectral Modeling Synthesizer

The goal in developing the real-time, spectral modeling, monophonic synthesizer was to use the audio descriptors extracted from the violin to control the synthesis process thereby allowing the performer more control over the synthesis process than would be possible using a MIDI keyboard. However, in order for this greater control to translate into greater expressivity in a performance, there must be a fairly intuitive mapping between the control information extracted from the violin and the control inputs of the spectral model.

The author chose to concentrate on modeling the sound of an ebow while developing the synthesizer. The ebow is a battery powered device used for playing guitar that manages to produce a sound similar in nature to that of a bowed string by producing an electro-magnetic field which causes the steel strings of a guitar to vibrate. By changing the ebow's position on the string, different string overtones can be produced, and fade-ins and fade-outs can be produced by lowering and raising the ebow from the string. It is fairly obvious how to map the descriptors extracted from the violin to the descriptors for the ebow’s spectral model, and the sound of an ebow differs enough from that of a violin to make it a welcome addition to a violinist’s sound palette. For this reason the author limited his efforts to modeling the ebow although it is intended to eventually develop this application into a general purpose synthesizer supporting all three of the previously mentioned spectral modeling techniques.

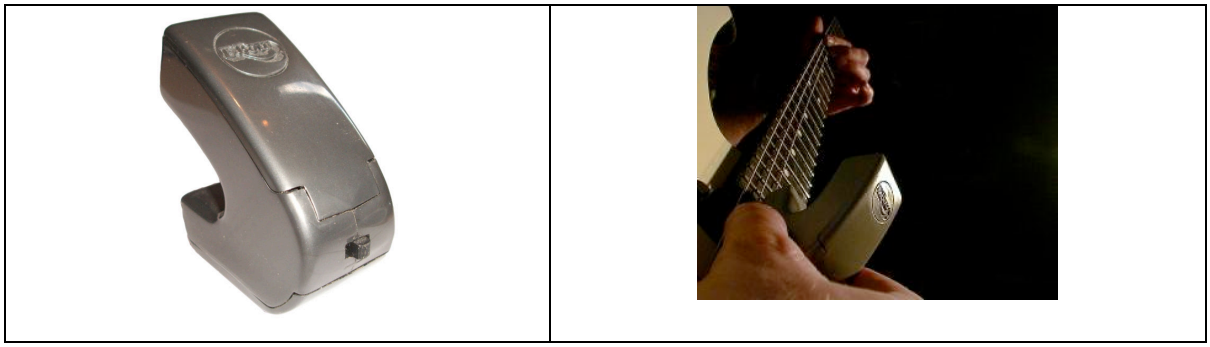


Figure 20 An ebow by itself and an ebow being used to play a guitar.

The synthesizer was developed in C++ using the CLAM library. The author used CLAM's classes for the implementation of sinusoidal plus residual modeling and SDIF file IO and developed additional classes for threaded, buffered file reading, thread pooling, looping, OSC input, data mapping, data management, spectral transformation and interpolation. The synthesis process is depicted in the following flow diagram.

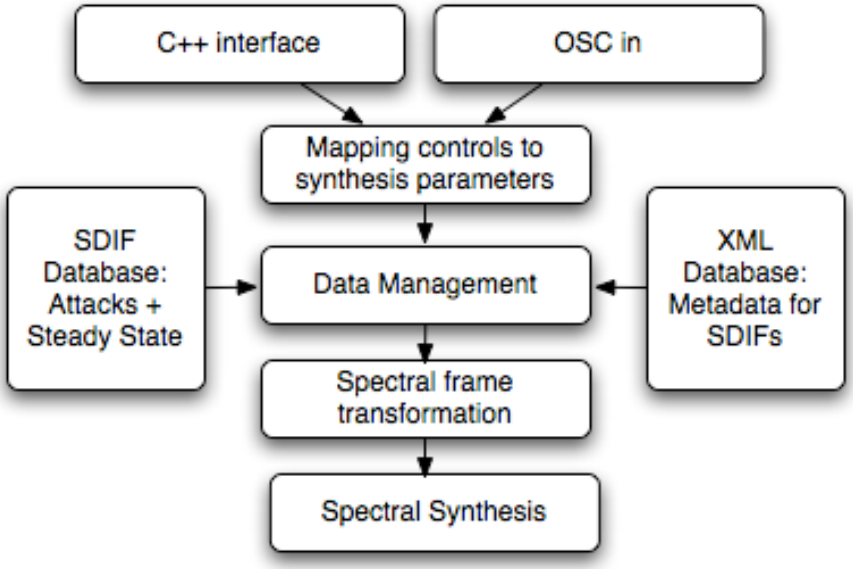


Figure 21 Overview of the synthesis process as implemented by the author

1. Input Sources

The synthesizer runs either as a module in CLAM's Network Editor

or as a standalone application that can be controlled either via a score file or an OSC stream. It is also planned to have it run as well as an external inside of Max/MSP, but at the time of writing this has not yet been completed.

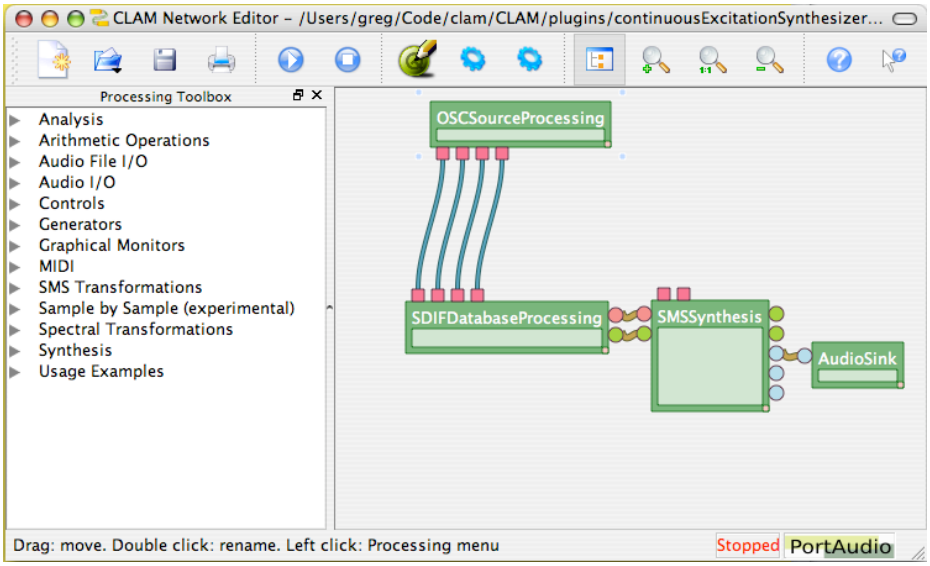


Figure 22 The synthesizer inside of CLAM's Network Editor

When loaded in OSC listening mode, the synthesizer listens on port 7000 for OSC events with the following syntax:

/ces f f f f pitch amplitude brightness voiceId

The message's intended recipient – the continuous excitation synthesizer – is given by the string "/ces". The following four "f" letters give the data types of the subsequent four variables; they are all floats. The variable "pitch" gives the pitch of the note in hertz. The variable "amplitude" gives the amplitude in the range from 0 – 1, and the variable "brightness" gives the brightness in the range from 0 – 1. The brightness value can also be set to -1 in which case the amplitude is used as an indicator of the brightness instead. Finally, the variable "voiceId" is used to indicate whether the events belong to an existing note / phrase or a new one. This is the means for clients to indicate where notes should be

segmented; for each new voiceId, the note begins with an attack phase while for existing voiceIds once the attack phase is completed additional frames are read from the steady state which loops indefinitely. It is preferable for clients to indicate the note segmentation points, because the clients are in a better position to make decisions about the proper note segmentation than the synthesizer, but client's can chose not to provide this information by setting the voiceId to -1. In this case the synthesizer segments the event stream into notes by looking for changes in the pitch value greater than a predefined threshold.

2. SDIF and XML Database

One of the primary goals when developing the synthesizer was to ensure that new audio material could be converted into a usable database format by musicians. This suggested the following constraints: the database creation process should require little to no knowledge of spectral modeling techniques, and the process should be quick, easy, and relatively fail proof.

To meet these constraints, the author developed a simple utility program for converting audio material into SDIF files with associated metadata in XML format. Before the analysis, the user specifies where the attack ends and how far into the steady state he would like to search for loopable audio. The program then performs a sinusoidal plus residual analysis on the file and searches for the optimal loop points amongst the analysis frames by comparing each frame using the earth mover's distance algorithm (Rubner, et al, 1998). This algorithm can be understood metaphorically by imagining the analysis frames to be representations of piles of dirt spread out over a region; the earth mover's distance is then the cost of turning one pile into the other where cost is defined to be the minimum amount of dirt that must be moved times the distance by which the dirt has to be carried. The frames with

the smallest earth mover's distance are the most spectrally similar, and they are therefore presumably the best loop points. After the program has extracted a half a dozen loops in this manner, it saves their start and end positions in milliseconds to an XML file together with the fundamental frequency, the maximum amplitude[3], and the start point of the attack. Finally, the program saves the analysis frames to a single SDIF file.

Currently, this utility program runs as a command line application, but ultimately, it would be desirable to integrate this functionality into the existing CLAM application SMSTools which provides a GUI interface to the sinusoidal plus residual analysis process in order to provide users with a better looking and eventually more powerful interface.

3. Data Management

When the synthesizer starts, it is given a directory name containing the database. The synthesizer then searches for files in this directory ending in the extension "xml" and after loading them converts them into metadata objects. It then opens file handles to the SDIF files referenced in the metadata objects and preloads the attack and first loop of each file. This manner of dynamic loading enables users to grow or shrink the database by simply moving files into or out of the directory.

Later when a file is played, the additional loops are loaded as well into memory by a background thread if the file is played long enough to start looping. This ensures a higher quality synthesis by preventing the same loop from being played contiguously while being conservative with regards to memory consumption.

[3] The maximum amplitude is extracted as a surrogate for brightness. Typically, notes that are played louder are brighter, and for this reason if the synthesizer identifies two notes with the same frequency in the database, it will assume that the note with the greater maximum amplitude is brighter. This information is then used when changing the equalization of the sinusoids in response to the brightness parameter.

4. Interpolation

The metadata for the SDIF files are indexed internally in a two dimensional table where the axis are pitch and brightness. As control information for pitch, amplitude and brightness comes in, the data manager must create a frame to match the target pitch and brightness from a potentially insufficient data set. As users can put an arbitrary number of files in the synthesizer's directory, they may very well choose to place only a single sample or a non-multilayered collection of samples in the directory rather than a multilayered collection of samples, and for this reason the interpolation algorithm must be flexible enough to deal both with the best case and the worst case scenarios in the optimal manner.

The interpolation algorithm implemented by the author chooses the SDIF data source with the smallest Euclidian distance from the target pitch and brightness as the base source for the interpolation. The pitch of its delivered frame is then transposed to match that of the target pitch. Then, a second data source is searched for which can be used as the other pole with which to interpolate the brightness. If a data source with a suitable brightness value is found that has a pitch distance which is equal to or less than that of the first data source or if a data source with a suitable brightness is found that has a pitch distance which is greater than that of the first data source but does not exceed five semitones, then it is selected as the second pole for brightness interpolation. The pitch of the delivered frame is also transposed to match that of the target pitch, and then, the brightness is interpolated by scaling the magnitudes of the sinusoids of the first frame with the envelope of the sinusoids of the second frame.

5. Synthesis

The interpolated frame is then given to CLAM's sinusoidal plus residual modeling class to be resynthesized. Before the frame is given over to the IFFT, the phase for each STFT bin is recalculated based on the frame's transposed frequency and the number of samples since the last frame, and the sinusoids and the residual are merged into a single spectrum. Then, the time domain signal is created with the IFFT, and the signal is windowed, overlapped, and added to the last window of the signal.

V. Conclusions and Future Work

An audio driven synthesizer was presented that was developed for a bowed string controller. Algorithms were given for detecting the pitch, brightness and bow direction of a violin and for identifying transients, and a real-time spectral modeling synthesizer was introduced that uses the descriptors extracted from the violin's signal to control audio synthesis from a spectral model.

In the future the author would like to continue refining and extending the audio descriptors for violin. With further work the standard deviation of the spectral peak slope descriptor could likely be further reduced. The transient detection algorithm could likely be further improved by developing better signal level descriptors; a dissertation written recently on the topic promises to provide a wealth of further insights and ideas in this direction (Thornburg, 2005). And it could be potentially useful to develop classification algorithms for the kind of string motion, i.e. Helmholtz motion, double-slipping motion, etc as well as the style of bowing, i.e. staccato, detaché, pizzicato, etc.

With regards to the synthesis engine, one could honestly say that the real work is only now beginning. The core classes for synthesizing from a spectral model in real-time are now finished, but as was mentioned in the introduction, the author is most interested in spectral models due to the variety of transformations of sampled material that they allow. The author would like to develop a flexible framework for specifying sound transformations using spectral models that would allow these transformations to be controlled in real-time with input from the violin possibly after they have been parametrized with contextual information regarding tempo and key from the surrounding musical environment.

VI. Bibliography

Amatriain, X., P. Arumí and D. Garcia. "CLAM: A Framework for Efficient and Rapid Development of Cross-Platform Audio Applications." Proceedings of ACM Multimedia, Santa Barbara, U.S.A., 2006.

Askenfelt, A. "Measurement of Bow Motion and Bow Force in Violin Playing." Journal of the Acoustical Society of America 80.4 (Oct. 1986.): 1007-15. .

---. "Measurement of the Bowing Parameters in Violin Playing." Journal of the Acoustical Society of America 84.1 (1988): 163. .

Benade, A. H. Fundamentals of Musical Acoustics. 2nd ed. New York: Dover Publications, 1990.

Chamberlin, H. "Using the FFT for Synthesis." Music Applications of Microprocessors.Hayden Book Co., 1980. 424-431.

Cremer, Lothar. The Physics of the Violin. Trans. John S. Allen. 1st ed. Cambridge, Massachusetts: MIT Press, 1984.

Disley, Alastair, and David Howard. "Spectral Correlates of Timbral Semantics Relating to the Pipe Organ." Baltic-Nordic Acoustic Meeting, Mariehamn, Finland, 2004.

George, E. B., and M. J. T. Smith. "Analysis-by-Synthesis/Overlap-Add Sinusoidal Modeling Applied to the Analysis and Synthesis of Musical Tones." J. Audio Eng. Soc. 40.6 (June 1992) .

Guettler, Knut. The Bowed String. KTH, Department of Speech, Music and Hearing, 2002.

Helmholtz, Hermann L. von. On the Sensations of Tone as a Physiological Basis for the Theory of Music. New York: Dover, 1885 (Reprinted 1954).

Jehan, T., and B. Schoner. An Audio-Driven Perceptually Meaningful Timbre Synthesizer., Proc. Intl. Computer Music Conference, 2001.

Kellum, Greg. "The Music Is Movement Plug-in Collection." 2005. <http://artassault.gregkellum.com/software/>.

Krishnaswamy, A., and J. O. Smith. " Inferring Control Inputs to an Acoustic Violin from Audio Spectra." Proc. Intl. Conf. on Multimedia and Expo, 2003.

Maher, R., and James Beauchamp. "Fundamental frequency estimation of musical signals using a Two-Way Mismatch procedure", J. Acoust. Soc. Am., Vol. 95, No.4, pp. 2254-2263. 1993.

McAulay, R. J., and T. F. Quatieri. "Speech Analysis/Synthesis Based on a Sinusoidal Representation." IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 34.6 (April 1986): 1713. .

Middleton, Gareth. "Pitch Detection Algorithms." 12/03/2003. <http://cnx.org/content/m11714/latest/>.

Moorer, J. A. "The use of the Phase Vocoder in Computer Music Applications." Journal of the Acoustical Society of America 26.1/2 (1978): 42 - 45. .

Rubner, Yossi, Carlo Tomasi and Leonidas J. Guibas. "A Metric for Distributions with Applications to Image Databases." ICCV '98: Proceedings of the Sixth International Conference on Computer Vision.

Schoner, B. Probabilistic Characterization and Synthesis of Complex Driven Systems. PhD MIT. 2000.

Serra, X., et al. "Integrating Complementary Spectral Models in the Design of a Musical Synthesizer." Proc. Intl. Computer Music Conference, 1997.

Serra, X. "Musical Sound Modeling with Sinusoids Plus Noise." Musical Signal Processing. Ed. G. D. Poli, et al.Swets & Zeitlinger Publishers, 1997.

Thornburg, Harvey. Detection and Modeling of Transient Audio Signals with Prior Information. PhD Stanford University, 2005.

Velikic, G., E. L. Titlebaum and M. F. Bocko. " Musical Note Segmentation Employing Combined Time and Frequency Analyses." Proc. IEEE Conf. on Acoustics, Speech, and Signal Processing, 2004.

Weisstein, Eric. "Least Squares Fitting." 8/24/2007 2007. <http://mathworld.wolfram.com/LeastSquaresFitting.html>.

Wessel, D., C. Drame and M. and Wright. "Removing the Time Axis from Spectral Model Analysis Based Additive Synthesis: Neural Networks Versus Memory-Based Machine Learning." Proc. Int. Computer Music Conference, Ann Arbor, USA, 1998.

Witten, Ian H., and Eibe Frank. Data Mining: Practical Machine Learning Tools and Techniques. 2nd edition ed. San Francisco: Morgan Kaufmann, 2005.

Woodhouse, J., and P. M. Galluzzo. "The Bowed String as we Know it Today." Acta Acustica united with Acustica 90 (July/August 2004): 579-589 (11). 2004.

Woodhouse, Jim, and P. M. Galluzzo. "Why is the violin so hard to play?" +plus magazine. September 2004 2004. <http://plus.maths.org/issue31/features/woodhouse/index.html>.

Woodhouse, Jim. "Bowed String Simulation using a Thermal Friction Model." Acta Acustica 89: 355–368. 2003.

Yoo, Lilit, and Ichiro Fujinaga. "A Comparative Latency Study of Hardware and Software Pitch-Trackers." Proc. SEAMUS, 1998.