INFERRING CONTROL INPUTS TO AN ACOUSTIC VIOLIN FROM AUDIO SPECTRA

Arvindh Krishnaswamy *Julius O. Smith*

Center for Computer Research in Music and Acoustics
Dept of Electrical Engineering, Stanford University
arvindh|jos@ccrma.stanford.edu

ABSTRACT

We present a method to analyze the streaming sound output of a real acoustic violin and infer the control inputs used by the musician. In particular, we extract the following parameters: which note was played, which string it was played on, whether the instrument was bowed or plucked and the location of the bowing or plucking point from a discrete set. The approach we use, pattern classification of the Short-Time Fourier Transform (STFT) frames of the sound signal, is general enough to enable application to almost any musical instrument.

1. INTRODUCTION

When analyzing a monophonic audio signal from a musical instrument, the traditional parameters estimated are pitch, loudness and timbre for transcription and synthesis purposes. Recently, however, there has been growing interest in extracting information about the control inputs used to play the instrument. For example, in [1] the plucking point of an acoustic guitar is determined, and in [2] a pattern recognition approach is used to invert a computer model of a bowed string.

In this paper, our goal is to analyze a real-world musical signal: sound output from an acoustic violin played by a musician. We would like to determine which note was played, which string it was played on and whether the string was bowed or plucked. Also, from a discrete set of allowed locations, we would like to determine the bowing or plucking point.

The reason we can extract such information is because a note with the same pitch would sound different (have a different timbre or spectral energy distribution) when played on different strings with different string excitation points.

Our approach is based on pattern learning and classification of the STFT frames of the musical signal. The STFT was chosen as the front-end due to its ease of implementation and relatively low computational cost compared to other spectral transforms. But most importantly, it is widely used in audio analysis due to its approximate correspondence to the function of the inner ear in hearing. Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA) are used for dimension reduction of the "normalized" STFT feature vector, and classification is performed using a simple Euclidean distance metric. The methods presented here are general enough that application to other musical instruments is simple and straightforward.

Applications of such technology are several. For example, knowledge of not only what was played but also *how* it was played may enable better sound analysis and synthesis algorithms and serve as a starting point for more accurate analysis techniques which may require a "seed" or "local region" to begin with. Also, if the control inputs can be inferred, a musical instrument may be used as a more general human-computer interaction device.

2. SPECTRAL CLASSES

We worked with 208 varieties (classes) of "spectral patterns," and we enumerate them in the following discussion.

We allowed a total of 13 notes (including the open string) to be played on each string of the violin, (thus spanning one octave on each string), and each note was a semitone apart from its neighbors. We labeled each note, in pitch order, with a number starting from 00. I.e., the lowest pitch note, which is the open string note on String 1, would be Note 00. Note that the same note can occur on multiple strings. For example, Note 12 can be played as an open string on String 3, or by stopping String 1 or 2 appropriately as shown in Figure 1 (which illustrates some other note positions also). Table 1 gives the note ranges on each string.

Each note could be played by bowing or plucking, and for each of these two actions, two different string contact positions are allowed, as depicted in Figure 1. We assign notes with the same pitch but played on different strings to different classes (so that we can determine which string was played). Since we have 4 strings, we get a total of $4 \cdot 13 \cdot 2 \cdot 2 = 208$ classes.

The previously stated restrictions of allowing only 13 notes per string or only two positions for bowing or plucking were just to limit the number of classes and simplify somewhat our initial experimental attempts. Our methods

In *Proc. IEEE ICME 2003, Jul 6-9, Baltimore, Maryland.*

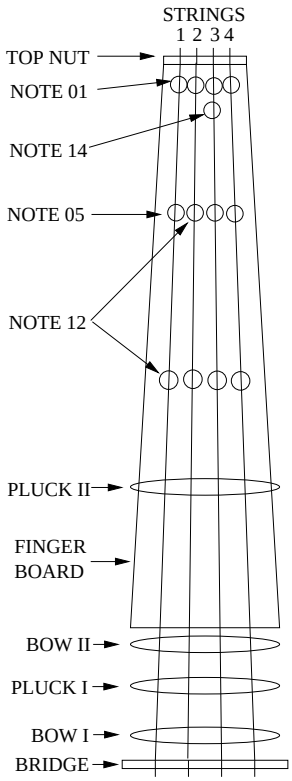

Fig. 1. The Violin Fingerboard with bowing, plucking and a few note positions indicated.

will work with any number of classes, notes per string and string excitation points.

3. EXPERIMENTAL SETUP

The violin used in our experiments was a 1984 "Karl Hofner" from Germany with medium gauge Pirastro Eudoxa strings which were tuned as shown in Table 1.

This alternate tuning system was used *not* due to any limitations in our analysis methods but rather only because the violin was also being used for other purposes for which this tuning scheme was most convenient. (Note that in this tuning, for example, the E-string is not really tuned to the pitch E).

The audio samples were recorded using a desktop PC soundcard and an inexpensive mic at a sampling rate of 24 KHz and 16 bits PCM quantization.

We recorded an example sound from each class for training our pattern classifier. For testing purposes, subsequent recordings were made of individual classes (notes) as well as typical real-life melodies.

String	Number	Pitch (Hz)	Note Range
G	1	173.50	00 - 12
D	2	260.25	07 - 19
A	3	347.00	12 - 24
E	4	520.50	19 - 31

Table 1. Tuning of the violin strings.

4. FEATURE VECTOR GENERATION

We now describe details as to how the audio signal from the violin is processed to generate a feature vector for classification.

First, the STFT is computed using a Hamming window and the following parameters: WindowSize = 20 ms, HopSize = 10 ms and DFTSize = 8192, which gives frequency bins separated by roughly 2.9 Hz given our 24KHz sampling rate.

The logarithm of the magnitude ($20 \log_{10} |\cdot|$) (dB) of each STFT frame is then computed, and the phases are discarded. Typically, frequency bins above 10KHz are also discarded. (Keeping more or less of the frequency spectrum could affect the results). Each frame is then normalized for loudness (roughly speaking) by subtracting out its maximum value. This sets the peak value in any frame to 0 dB. Finally, all values below -27 dB are clipped to -27 dB, which gets rid of the sidelobes of the window and other noisy, unimportant features, leaving frames/vectors that contain only the most prominent spectral peaks. We shall call the i-th (column) vector in this set v_i.

The dimension of this vector is very large. If we consider 0 - 10 KHz, for example, there are 3413 frequency bins. To reduce its dimension, PCA and LDA are used similar to [3].

The matrix of PCA column vectors, W_p, and the corresponding diagonal matrix of eigenvalues, D_p, are obtained by solving an eigenvalue problem:

$$R \cdot W_p = W_p \cdot D_p$$

where R is the covariance matrix of the data:

$$R = E[(v_i - \overline{v})(v_i - \overline{v})^T]$$

The LDA vectors and eigenvalues, W_l and D_l, are obtained by solving a *generalized* eigenvalue problem:

$$S_b \cdot W_l = S_w \cdot W_l \cdot D_l$$

where S_b and S_w are the "between-class" and "within-class" scattering matrices of the data [4, 5].

We should point out that the only reason we use PCA in our method is to make the LDA computation more stable. Thus, we retain most of the PCA coefficients, discarding

only a few unimportant ones (corresponding to eigenvalues that are virtually zero). Almost all of the dimension reduction is achieved by LDA.

With m classes, LDA yields up to $(m-1)$ projection vectors. We then have the choice to keep any number of them, and we discuss the results of using various numbers of these projections vectors later. Let N_v be the dimension of v_i, and let N_p be the number of LDA projection vectors we keep.

The PCA and LDA projections are combined into one (N_v x N_p) projection matrix, $W = W_p \cdot W_l$, which is applied to v_i to yield the final feature vector, $f_i = W^T v_i$.

Figure 2 illustrates the steps involved in generating the feature vectors f_i from the audio samples $x[n]$.

The preceding steps were implemented in Matlab. The STFTs were computed using the `fft` function and the PCA and LDA vectors were determined by solving eigenvalue problems using the `eig` function.

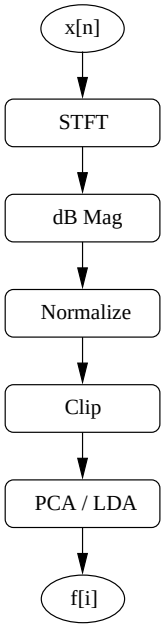

Fig. 2. Generation of the feature vectors.

5. TRAINING AND CLASSIFICATION

In the training stage, we assembled several databases, each containing a different subset of the 208 classes listed before. For example, the "main" database included all 208 classes, another contained only plucked notes, and yet another included all instances of the Notes 00, 12 and 24 only.

The reason we do not automatically include all 208 classes into one database and leave it at that is because it may be better to have a hierarchy of databases, each adding a marginal piece of information about what was played. This is because LDA may do a better job of separating classes which are close together if it only has to deal with a smaller subset of all the classes.

For each database, we generated feature vectors for the classes it contained using the recorded sound samples, but stored the class means only. It should be noted that the projection vectors W_p and W_l associated with each database are determined by which classes are present. Parameters like N_v and N_p have to be fixed within each database, but we are free to vary them as we create different databases.

We also formed "super-classes" by combining one or more classes. For example, all plucked notes could be merged into one class, and all bowed notes into another. A database with these two "super-classes" would be able to distinguish whether a note was bowed or plucked, but nothing more beyond that.

Within a given database, classification of an unknown STFT frame is performed by first generating a feature vector using the appropriate projection matrix and by then finding the closest match from the class means using Euclidean distance: $d(f_i, f_k) = |(f_i - f_k)^T (f_i - f_k)|^{1/2}$.

6. RESULTS AND DISCUSSION

In our experiments we found that, when using our largest database (with 208 classes), the classification system was able to identify correctly which of the 32 pitches was played at any time. However, the estimates of the other parameters were not reliable, as illustrated in Figure 3. In this test case, the sound consisted a series of notes bowed (in position I) on strings 3 and 4. In the top of this figure, a portion of the (normalized and clipped) STFT spectrum is also shown.

But if we are able to recover the pitch correctly, we may resort to a smaller database focusing only on that pitch. For example, if we are sure that one of Note 00, 12 or 24 was played, then we could use a database with only those classes. As shown in Figure 4, this yields much better results even though a narrower portion of the frequency spectrum was used (125 - 6000 Hz instead of 0 - 10 kHz). In this case, the test sound consisted of concatenated 0.5 sec sound segments, each a different variation of Note 12. It should be clear from the plots how each segment was played. N_p was 23 in this case. Incidentally, notes 00, 12 and 24 are from the same chromatic class but in different octaves. Therefore, even if we have initial octave errors due to our pitch tracker (or our "main database"), the correct octave can still be determined eventually.

In Figure 5 the results were obtained with only 5 LDA projection vectors. Though performance has degraded, it is still reasonably good overall. The most significant errors occur when plucking Note 12 on String 1. This is because (see Figure 1) the two plucking points are at very similar

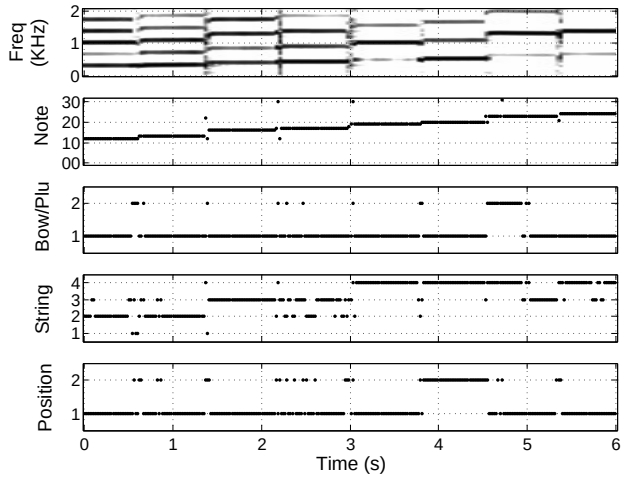

Fig. 3. Results with all 208 classes in database.

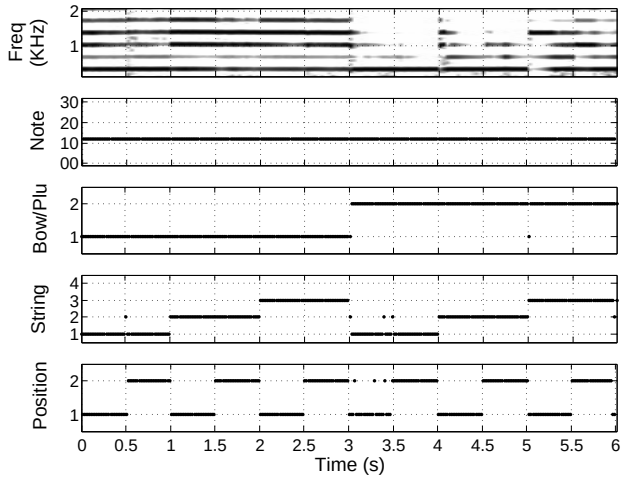

Fig. 4. Results with only Note 00, 12 & 24 classes in the database and with a narrower portion of the frequency spectrum used.

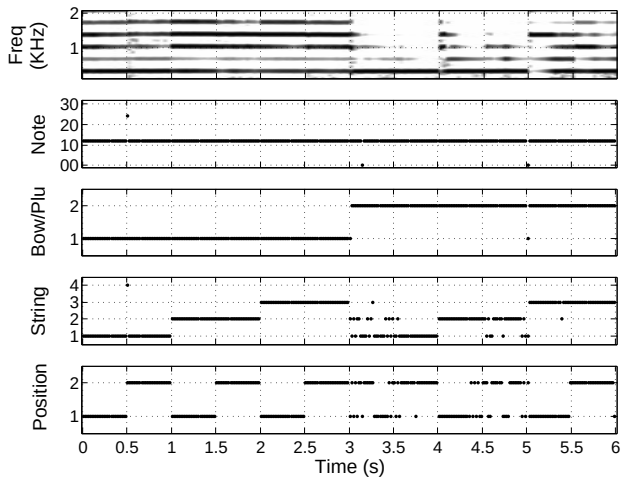

Fig. 5. Same as database used to produce Figure 4, but with only five LDA projection vectors.

locations with respect to the two endpoints of the vibrating string when Note 12 is played. I.e., their spectral characteristics are also very similar. Thus, it is not easy to separate these two classes to begin with and especially so with only 5 projection vectors. For a human classifier this task was difficult too.

As can be seen in the graphs, there are "errors" that occur at transient locations. However these errors could be handled by the introduction of additional classes for transients. Also, the spectral shapes of some sounds change drastically over time, especially in the plucked note cases. To address this difficulty, certain classes may be broken up into sub-classes across time, relative to note onset.

Future work will include expanding the number of notes and bow positions allowed. It may also be possible to determine such variables as bow speed and magnitude of each pluck. We may, in addition, introduce classes of double stops (two notes sounded at once). Also, it will be interesting to test a database that was created using a particular violin with sounds from other violins. Finally, results obtained using our algorithm could be compared systematically with the capabilities of a well-trained musician. Doing so may offer insights which could improve our technique.

Overall, our approach has been very successful. Implementation of the algorithms is easy, but parameters such as N_v and N_p as well as the division of the classes into various databases will need to be fine-tuned for performance and computational efficiency.

7. REFERENCES

[1] Traube and Smith, "Extracting the fingering and the plucking points on a guitar string from a recording," in *Proc IEEE WASPAA*, 2001.

[2] Serafin, Smith, Thornburg, et al., "Data driven identification and computer animation of bowed string model," in *Proc ICMC*, 2001.

[3] Zhao, Chellappa, and Krishnaswamy, "Discriminant analysis of principle components for face recognition," in *Proc 3rd Intl Conf on Automatic Face and Gesture Recognition*, 1998.

[4] Fukunaga, *Introduction to Statistical Pattern Recognition*, Academic Press, 2nd edition, 1990.

[5] Duda et al., *Pattern Classification*, John Wiley and Sons, New York, 2nd edition, 2001.