An Interface for Real-time Classification of Articulations Produced by Violin Bowing

Chad Peiper†, David Warden†, Guy Garnett‡
†Department of Computer Science, University of Illinois Urbana-Champaign
‡School of Music, University of Illinois Urbana-Champaign
{peiper,warden,garnett}@uiuc.edu

ABSTRACT

We introduce a software system for real-time classification of violin bow strokes (articulations). The system uses an electromagnetic motion tracking system to capture raw gesture data. The data is analyzed to extract stroke features. These features are provided to a decision tree for training and classification. Feedback from feature and classification data is presented visually in an immersive graphic environment.

1. INTRODUCTION

The CyberViolin project seeks to capture aspects of violin playing in order to better understand how expert violinists achieve the effects they are heard to achieve. Ultimately, this information will be used to facilitate human-computer interaction.

There have been several initiatives involving the use of sensor data to describe musical performance. In [8], a model is described for the mapping between gestural primitives and performer processes. A partial survey of instruments created which allow sensor data to manipulate the sound characteristics produced is given in [6]. Virtual musical instruments provide a gestural interface for the mapping of movement to sound [7]. In the eviolin project, a synthetic instrument is created, efficiently mapping gestural data to the production of particular sounds [9]. In [4], resonance model filter parameters are changed based on motion sensor data, allowing an instrument to be played through the resonance model of another sound. An instrument model allowing alternate input devices is described in [10]. In contrast to systems using sensor input to map gestures to sound, the cyberviolin project records gesture data for analysis in order to provide objective feedback to the user. The user may then interactively adjust his/her performance.

Bow movement is one significant part of violin playing. Characteristics of this gesture include bow pressure, velocity, and position. Used together, these parameters result in different articulations; controlling these characteristics is essential to the production of good violin tone.

In order to analyze these properties a module allows the computer to identify and measure different articulations of bow strokes. Additionally, this technology is designed to be adapted to a pedagogical mode. Ideally, a violinist in an immersed virtual environment would be able to interact with a virtual world that responds to the articulations of the player. As a pedagogical tool, a violinist in the practice room could be monitored by a virtual teacher who could identify incorrect articulations in real-time.

2. DESCRIPTION OF THE SYSTEM

The software system for the classification of violin bow strokes (articulations) uses an electromagnetic motion tracking system with two sensors. One sensor is attached to the back of an acoustical violin and the other is attached to the frog of the bow. There are a variety of articulations that have developed historically as aspects of violin performance. These include the following: *détaché, martelé, staccato, spiccato*, and *legato*. The difference between these various strokes is the result of subtle variations in gestures, which are discrete and measurable.

Détaché is the most common bow stroke in string playing. It is usually played in the middle part of the bow, with one note per stroke. Although the spelling of the term suggests a detached articulation, the bow never stops between strokes. *Legato* is defined by the slurring of two or more notes on one bow stroke. Contrary to the continuous bow motion of the *legato* and *détaché*, *martelé* and *staccato* require decisive attacks and releases.

Martelé is characterized by a percussive stroke produced by pinching the string with the bow before drawing the bow. The pinching or *martelé*-accent results from the weight of the bow completely resting on the string with additional weight applied by the relaxed, hanging arm. In each stroke, almost all of this weight is suddenly released from the string, but only for a moment while the bow is in motion. Upon completion of a quick stroke (bow speed), the weight of the bow and arm is again pressed against the string resulting in discontinuity between strokes.

The *spiccato* is performed by a natural bouncing of the bow from the string in the middle portion of the bow. The exact location varies from bow to bow dependent on the location of the "bouncing point" on the stick. Performing an accelerating *détaché* in the middle-third of the bow will result in *spiccato*. Note that both *détaché* and *spiccato* are performed with a continuous bow motion (i.e., no stoppages).

The term *staccato* denotes two or more short strokes on the same bow; performing two *martelé* strokes on the same bow results in a *staccato*.

The CyberViolin system for analysis of bowing techniques consists of the following components: bow stroke characterization, decision tree induction, and bow stroke classification. Decision tree induction allows classification rules to be easily generated and compared with expectations from domain experts.

Bow stroke identification involves a representation of each stroke in terms of a dataset that translates physical activity into various classes to facilitate computer recognition. The next step is the creation of a tree structure that enables the computer to recognize the various streams of data that constitute the successful performance of each stroke. The final step is the sifting of the data through the tree, a process that results in the real-time identification by the computer of performance gestures.

The system can be used in two modes: training and classification. The two modes of operation are illustrated in the following figures:

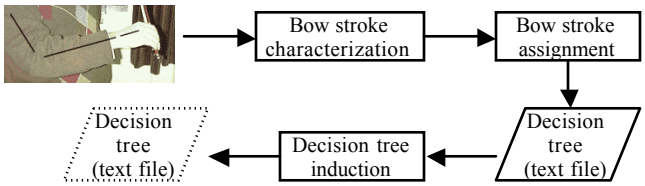


Figure 1: Training process

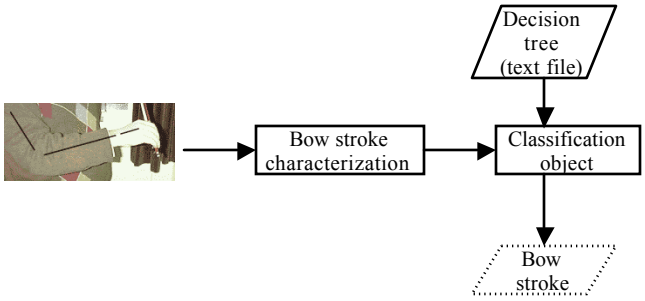


Figure 2: Classification process

The training mode is used to construct the decision tree, which the computer will use in the real-time classification. In order to facilitate the computer's ability to classify different articulations, a series of datasets that characterize each stroke must be provided. This is an example of supervised learning in that the programmer must provide the definitions for each aspect of a given stroke. This is the necessary prerequisite to enabling the computer to assess any given input. The more training data we provide, the greater the accuracy of the classification. In addition, there are algorithms designed to improve the structure of the tree. For example by combining branches we can create a more compact structure, and by removing unwanted branches created as a result of noise in the dataset, we can decrease the depth of the overall tree. In the classification mode, we traverse the tree to identify a class label.

Decision trees provide a natural mapping from feature data to classification rules. Unlike other real time classification methods, such as neural networks, these rules can be compared with expectations from domain experts. Since the decision trees generated in our system are relatively small, they can be easily modified to further simplify the tree and improve the classification accuracy. One of the major disadvantages of the decision tree approach is that tree induction algorithms have a poor scalability in terms of both the size of the training set and the number of attributes. While additional data would be beneficial, the relatively small number of attributes under consideration reduces the requirement for large data sets. In addition we have the option of manually editing and adjusting the tree as we see fit.

3. IMPLEMENTATION OF THE SYSTEM

3.1 Bow Stroke Characterization

For the purposes of our classification system we define a bow stroke as the path the sensor on the bow travels between consecutive changes in bow direction. The most fundamental classification of bow stroke is "up-bow" or "down-bow". Since the violin sensor is mounted along the axis of the strings, when the bow-sensor, mounted near the performer's hand, approaches the strings the distance between the two sensors is small. Therefore, an up-bow is defined by a decrease in the distance between the bow and violin sensors, while the down-bow is the reverse. Whenever movement is detected in the direction opposite of the current stroke in magnitude greater than a threshold of 0.4in a bow change is detected.

Initially the parameters used to characterize the bow strokes were: the distances between the two sensors at the beginning (D1) and at the end (D2) of the stroke, the length of the path of the bow sensor (L) computed from D1 and D2, and the average speed of the sensor (V). The most accurate measurement possible, given this sparse set of parameters, was approximately 73%.

Adding additional characteristics of bow movement has increased the accuracy of this evaluation: frequency of bow change, acceleration or deceleration within a stroke, continuity of motion between strokes, bow position (middle, upper, lower), number of changes in a single coordinate (stroke similarity), lack of movement within a stroke (stoppage). The system's design allows for the easy addition and removal of characterization parameters.

In order to add additional features, code specific to that characteristic must be written. Several articulations must then be performed in order to record how this characteristic varies among articulations. Once this is done, numerical or nominal data can be presented to the decision tree program. The ability of the additional parameter to distinguish among articulations under consideration can be determined and can then be evaluated. An increase in accuracy indicates the addition was successful.

Other parameters were considered but did not lead to an increase in accuracy during our experiments. These characteristics included the change in azimuth, elevation, and roll relative to the violin.

The bow stroke characterization module is implemented in an ANSI-C++ program using the FreeVR library [2]. The library provides a programming interface to more easily obtain calibrated sensor data and render three-dimensional objects. The data is gathered from position sensors on the bow and violin in the NCSA CAVE™ Autonomous Virtual Environment [11].

The CAVE™ consists of a 10'x10' area with images projected in front, to both sides, and below the user. Shutter glasses allow images to appear in three dimensions. The special tracked glasses and CAVE™ wand are both attached to electromagnetic sensors that allow the computer system to know where within the CAVE™ each is located and what its orientation is. It supports the use of pulsed DC magnetic sensors which report position as well as azimuth, elevation, and roll. The environment is free of metal objects that would reduce sensor accuracy.

The raw position data can be written to a text file for later analysis. The feature data is also sent to a file (training mode) or sent to the classification object (classification mode). Each row in the file created as part of the training mode is a comma-separated list of decimal numbers that characterize a single bow stroke. The user must identify (supervised learning) each type of bow stroke manually at the beginning of each file of the training set.

3.2 Decision Tree Induction

The decision tree induction module is a stand alone C++ program, which, given a text file containing the training dataset, builds a decision tree and stores it in a separate text file. The bow stroke classification object accepts the file as input and uses the decision tree to classify the bow strokes. The decision tree induction algorithm is a version of the classic ID3 algorithm, which works only with discrete values and creates very shallow trees (the original ID3 creates binary trees and works with continuous values). The shallow decision

trees have very high classification performance and are easier for manual modifications. The algorithm is described in detail on page 56 in [1] and page 285 in [2]. "Information gain" is used as the attribute selection measure. Because this algorithm only works with discrete values, the range of values for any given stroke must be precisely defined before running the algorithm. Entropy-based discretization [1] [2], is used because it works best with the decision tree induction algorithm being employed.

With each iteration, the samples are split across value of the attribute which results in the least randomness among the resulting partitions. The process is repeated until the attribute list is exhausted or until a partition has only members of a single class. The nodes are labeled with the most common class present in that partition.

The attributes are selected from among the extracted features (such as velocity), and the classes correspond to the different articulations.

The command line parameters of the decision tree induction program are:

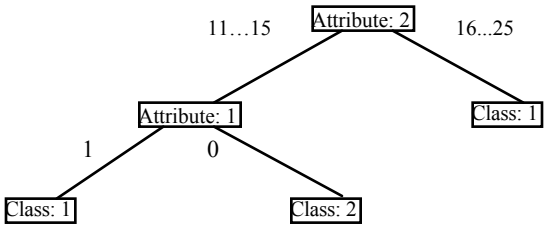
dtree input_file [output_file] [delta]

- input_file is the file containing the training set.
- output_file is the resultant decision tree exported by the function.
- delta represents a decimal number between 0 and 1 which is used in the stopping condition of the entropy-based discretization. The smaller the value, the smaller the ranges created.

The program evaluates the classification accuracy. 2/3 of the training set is used for decision tree induction and 1/3 is used to test the classification accuracy of the constructed tree.

3.3 Bow Stroke Classification Object

The bow stroke classification object (figure 3) acquires the extracted feature data and processes it through the decision tree. The classification object then returns the corresponding articulation. Upon instantiation, the classification object is initialized with a text file containing the decision tree. Each row in the file represents one node (figure 3). This module is implemented in C++ and is integrated with the bow stroke characterization code.




```
Attribute: 2
11…15: Attribute 1
          1: Class: 1
          0: Class: 2
16…25: Class 1
```

Figure 3: Textual representation

4. FEATURE DETECTION PROCESS

In order to gather the required data, two sensors are used in a CAVE™ Autonomous Virtual Environment. Each sensor provides information on its location in 3 dimensional space as well as the azimuth, roll and elevation to which it is oriented. In the CAVE™, this information is reported approximately 30 times per second, with an accuracy of approximately 1/4in.

Once the information has been captured from the sensors it is translated from the absolute coordinate system into a system relative to the location and orientation of the violin. This allows the bow movement to be correctly recorded and compared even as the user moves the violin about the area. If the bow is not within range of the violin, data is not recorded.

Raw data is recorded in a temporary data structure until an entire bow stroke can be analyzed. Since a stroke may stop in the middle, as in the case of a *staccato*, a stroke is not delineated until the next stroke is begun. This is determined from movement in the opposite direction beyond a predetermined threshold sufficient to account for any unintentional movement and sensor jitter.

Most basic feature information is determined at the stroke level. This information includes the bow direction (up or down bow) velocity, minimum and maximum position in each dimension, the time in the stroke when respective minima and maxima are achieved, the number of times the bow is stopped during the stroke, continuity, and acceleration. Acceleration is recorded for several discrete segments of bow used in any particular stroke. A violin simulator that reads in raw position data from a file aids in feature development.

The features of several bow strokes are also recorded in a temporary structure. Additional features can be reported based on the similarity of stroke properties over a sequence of several strokes. Strokes are also paired together before being sent to the decision tree for classification.

As there are limitations on the sensor and characterization system, the attribute characteristics may differ to some degree from expectations of characteristics provided by experts in the domain. Experiments were conducted to determine the benefit of various attributes by examining the performance of the decision tree when these attributes are considered.

Once information for all features is gathered, those features or derivations, which provide the most information, are selected for the decision tree. Those features typically include, the velocity, change in distance between the frog of the bow and plane of the violin strings, number of stoppages, continuity, acceleration for first and middle segments, and similarity of bow usage. A conversion program takes the features and arranges the output as a list of comma-separated values.

The decision tree generator takes a set of sample data and produces a hierarchical system for classifying a stroke into an articulation based on the values of provided attributes. The accuracy of classification (based on withheld data) on data sets containing several different articulations is shown in Figure 1. While some of the data sets have modest accuracy, there is an explanation for the results. When only *détaché* and *martelé* articulations are considered, 100% accuracy is obtained. The attributes considered by the decision tree help to discriminate between these articulations. The essence of *détaché* is that the bow never stops between strokes (continuity) whereas *martelé* contains discontinuity.

All articulations have characteristics that are derived from either the *détaché* or *martelé* stroke. For example the *legato* and *spiccato* are both continuous in the manner of *détaché* whereas the *staccato*, as mentioned earlier, is essentially multiple *martelé* stokes contained within a single bow.

There are two problems that result from such similarity of articulations. First, the decision tree can misclassify similar articulations when devoid of distinguishing attributes; it also requires a greater amount of data to differentiate articulations that share common attributes. Second, since a stroke is defined as a change in bow direction, it is difficult to take advantage of distinguishing attributes within a single stroke. It is also

difficult to consider attributes aggregated over several strokes such as the number of bow changes over time. This results in an insufficient amount of data available to the decision tree.

Staccato should be identifiable in a single stroke; the number of stoppages uniquely distinguishes this articulation. However, when several elements of a *staccato* articulation are aggregated to the point of a bow change, it appears similar to a *martelé*. Likewise *détaché* and *Legato* overlap with the exception that there will be fewer *Legato* strokes over time, which is only reflected to some degree in the velocity. Multiple stroke attributes such as examination of the similarity of bow usage among strokes increases accuracy, however the amount of data required by the decision tree is also increased. Given these constraints and the ability of the system to identify the two most general articulations the results are reasonable.

Table 1: Articulation classification accuracy

Articulation	Accuracy
martelé, détaché	100%
martelé, détaché, legato	85%
martelé, détaché, spiccato	81%
martelé, détaché, staccato	76%
staccato, spiccato	75%
martelé, détaché, spiccato, legato, staccato	71%

Once a decision tree is generated, it can be supplied to the violin program. Given the same set of attributes supplied to the decision tree generator, the classification routine will provide the classification reported by the tree. If desired, the tree could be modified by hand. The reported classification can be used by visualization code (see section 5) or logged into a file. In order to reduce false reports of articulation changes, the reported classification is an average over the five most recent individual stroke classifications preformed by the decision tree.

The feature extraction process only requires simple arithmetic operations be performed whenever new sensor data is received. An advantage of decision trees is their high performance after the training process is complete. Each bow stroke only requires a number of comparisons at most equal to the depth of the tree, which is in turn at most the equal to the number of distinct features extracted. For these reasons, all computation can be done without observable performance latency. In the CAVE™ new data was received thirty times per second, and strokes would be completed after several seconds depending on the articulation.

5. CYBERVIOLIN APPLICATION

The cyberviolin application provides a player an environment for the visualization of performance data. The program reports feature data in real time. With a sufficient number of strokes, real-time classification data is also available. All data is archived for later presentation or analysis.

Currently the application offers two modes of operation: record and playback. In either of these modes, graphical feedback is available. For each attribute the respective graph illustrates the value with its corresponding bow stroke. This allows the user to observe how the attributes, interrelated, form articulations.

Static graphs representing pre-recorded data, while informative, are of no real benefit to the player during the actual performance. The aim of this application is to provide the user with real-time feedback. Providing this feedback in the CAVE™ environment creates a virtual interactive mirror allowing the performer to see him/herself objectively. The additional insight afforded in such a system allows the user to adjust and improve their performance in real time.

As an example, consider a user intending to perform a passage with a *spiccato* articulation. If the player mistakenly performs a *détaché* we would like the system to provide feedback to communicate this error and the means to correct it. The report of the detected articulation indicates an error but is devoid of the details necessary to influence a change. Presentation of the distinguishing attribute values to the user can alert the user to the requisite corrections.

The application is currently under development and we hope to report on later versions supporting greater interaction.

6. LIMITATIONS AND FUTURE WORK

There are several sources that limit the accuracy of this system. One weakness is the limited amount of information that is available from a single pair of position sensors. If sound analysis data or pressure sensors on the bow were available, they may provide valuable information for feature extraction and subsequent classification.

The precision of the hardware sensors as well as their refresh frequency also has a limitation on the ability to detect features. In testing the CAVE™ environment the precision was clearly superior to other environments with less sensor precision. Sensor error in such environments made feature detection substantially less accurate and difficult to develop. Further improvements could allow for more precise detection of stoppages, discontinuity, and shorter strokes, as they are most dependent on sensor precision. The size, weight, and positioning of the bow sensor challenge the player's ability to perform articulations. A smaller, lighter, wireless sensor would result in a more natural performance, and perhaps more distinct articulations.

Feature detection could be improved with additional identifying features. Features that compare several strokes are one area of development that may lead to improved accuracy. By computing information over an aggregate of strokes, the time period for information provided to the decision tree is increased. This allows for increased accuracy, as an articulation may not be evident for several strokes. Reliance on these features would require articulations long enough to provide data on groups of strokes.

In our current implementation, some feature information is not entirely accurate. Whenever there is a discontinuity or stoppage, there is some period of time when the system is uncertain if playing has resumed due to the time lag used to determine motion. This appears to lead to observable error in some instances correlated with stoppages. One such articulation is the *martelé* stroke, which is generally defined as a percussive bow stroke produced by "pinching" the string with the bow before starting the stroke. When detecting bow changes where the bow is stopped at the end of the stroke, the velocity values were not consistent on the up and down-bow, even when the bow speeds of both strokes were performed similarly. The exact effect of this error on the classification is unknown. It may be possible to reduce this error with more accurate sensors in the future.

The availability of more data would aid in the construction of the decision tree. Since the tree must speculate on unknown information, the more performances the tree is exposed to during training, the more accurate it will be. Spurious or unusual data would also be factored out. While a decision tree is capable of very accurate classification, other AI based

classification methods, such as neural networks could also potentially be used.

The user interacts with the program by using the bow in place of the traditional wand pointing device. A graphical interface is projected within the CAVE™ allowing the user to select desired options by directing the bow at the respective location.

Even with future improvements, there will be a level of classification beyond which additional accuracy is difficult to obtain. This is due to the fundamental properties of the underlying music, and limitations in the performer's ability. However, even with these limitations, the cyberviolin project is able to perform classification with a useful level of success.

ACKNOWLEDGMENTS

Our thanks go to Camille Goudeseune, Integrated System Laboratories (ISL), and William Sherman, NCSA, at the University of Illinois at Urbana-Champaign.

REFERENCES

[1] Mitchell T. Machine Learning. ISBN 0-07-042807-7, McGraw-Hill, 1997.

[2] Han J. and Kamber M. Data Mining: Concepts and Techniques. ISBN 1-55860-489-8, Morgan Kaufmann, 2000.

[3] http://www.freevr.org

[4] Garnett G., Goudesuene C., Johnson T. "Resonant Processing of Instrumental Sound Controlled by Spatial Position" to appear, 2003.

[5] Trueman, D., P. Cook. "BoSSA: The deconstructed violin reconstructed", Proc. 1999 Intl. Computer Music Conf. San Francisco: Computer Music Association 232-239.

[6] Trueman, D. "The Infinite Virtual Violin:" in Reinventing the Violin, http://www.music.princeton.edu/~dan/rtv/, Chap 3.

[7] Goto, S. "Virtual Musical Instruments: Technological Aspects and Interactive Performance Issues", Trends in gestural control of music, ed. M. Wanderley and M. Battier, Paris: Ircam Centre Pompidou, 217-230.

[8] Choi, I. "Gestural Primitives ad the context for computational processing in an interactive performance system", Trends in gestural control of music, ed. M. Wanderley and M. Battier, Paris: Ircam Centre Pompidou, 139-172.

[9] Goudesuene, C. Composing with Parameters for Synthetic Instruments, Ph.D. dissertation. Dept. of Music, University of Illinois at Urbana-Champaign, 2001.

[10] Serafin, S., Dudas, R., Wanderley, M., and Rodet, X. "Gestural Control of a Real-Time Physical Model of a Bowed String Instrument", Proc. Intl. Computer Music Conf, 375–378, 1999.

[11] http://cave.ncsa.uiuc.edu/