BOWING MODELS FOR STRING PLAYERS

Cordelia V. Hall, John T. O'Donnell, Bill Findlay

University of Glasgow
Department of Computing Science, Glasgow, UK
jtod@dcs.gla.ac.uk

ABSTRACT

A bowing is a sequence of bow motions that enable a piece of music to be played on a string instrument with an appropriate interpretation and sound. Traditional notation shows only the bow direction for a few notes. We propose a *bowing model*, which contains information about the bowings of all the notes, and we show how the bowing model can be represented in software and describe how algorithms can use it to perform several tasks that help the performer. The model, and the algorithms, are suitable both for offline editing of music and for presentation on an electronic display during performance. In particular, software can show or hide bowings on various notes, according to the performer's needs; it can calculate a full bowing; it can modify a bowing based on preferences indicated by the performer; and it can allow bowings to be archived and searched. This approach is not prescriptive: the performer is in full control of all artistic decisions, while the software carries out repetitive tasks.

1. INTRODUCTION

Bowing is one of the most difficult aspects of playing a string instrument, such as the violin, viola, cello, and double bass. In addition to the physical difficulties of controlling the bow, the player must work out a practical sequence of bow motions that enables a piece to be played and that leads to the right kind of sound. Such a sequence is called a *bowing*. A bowing specifies at least the direction the bow is moving for each note, and may also specify they type of bow stroke, how long the stroke is, which part of the bow is to be used, and the distance between the bow and the bridge.

There are two aspects of a bowing: the choice of a suitable set of bow motions while satisfying a set of constraints, and notations in the printed music that help the player to remember the bowing. Normally the printed notations indicate only a small part of the bowing explicitly, because excessive notation makes the music harder to read.

The problem of bowing is complicated because it combines musical interpretation, calculation, notations for recording decisions, communication, and negotiation with other players. String players often work out bowings by trial and error informed by experience. Some of the bowings are then written into the music, because they are often hard to remember. Furthermore, there may be external constraints. In many orchestras all the members of a string section are supposed to use the same bowing, and in chamber music the different parts should use compatible bowings.

Planning how to use the bow is important, and string players give it a lot of thought. Unfortunately, they seldom write those plans down, so other players cannot benefit from what they have learned.

Our hypothesis is that computers can help benefit string players by assisting them in solving practical bowing problems. To achieve this, we introduce a bowing model as a complete description of a bowing, which can be represented as a data structure in a computer program, along with algorithms that manipulate the bowing data structure. The remaining sections of this paper define what a bowing model is, present heuristics for calculating a bowing, describe an algorithm that implements the heuristics, and discuss the results of applying the software to several pieces of music.

2. BOWING MODELS

String players often use occasional annotations, such as up-bow or down-bow symbols, to remember how to play certain crucial notes. The bow directions for other notes can be worked out on-the-fly, based on experience. However, there are several useful tasks a computer can carry out to assist the performer, which require the computer to know how every note will be bowed.

Therefore we require a complete description of bow motions, including the direction the bow moves for every note, the part of the bow being used, and any other relevant information. We call this description a *bowing model*. Although the model defines the bow direction for every note, it would clutter the printed music to include bowing annotations for every note. Therefore only a small portion of a bowing model is presented to the performer, and it is useful to distinguish between a bowing model (represented as a data structure in a computer program) and its presentation in the printed music (indicated by annotations such as up and down bow symbols).

A bowing model must at least specify the bow direction for each note. It may also contain additional information,

such as the position of the bow between the tip and frog, the location of the contact point (near the bridge or near the fingerboard), the bow speed, etc. In this paper, we discuss bowing models that specify bow direction and bow position, but not bow speed.

3. COMPUTER SUPPORT FOR STRING PLAYERS

A computer can support a performer by providing a flexible presentation of the music, including the bowing model. As the player becomes more familiar with a piece, a smaller number of notes may need bowing marks. In a chamber music rehearsal, it is often useful to indicate the bow direction at unusual places (for example, at the beginning of each line) to make it easier for the group to start playing at an arbitrary point. Using either a computer with printer, or a flat-screen display on the music stand, the system can update the presentation of the music quickly and easily.

Another area where the computer can help is in calculating the details of a bowing, given some constraints. The aim of this approach is to leave the artistic decisions to the string player, leaving the computer to carry out the mundane and repetitive tasks. For example, a bowing might specify that a note is to be played upbow while it also causes the bow to be at the frog—an impossible situation. An algorithm can detect problems like this, and either highlight them for the player's attention, or reject the bowing and calculate an alternative one. However, the player should be in control. If he or she dislikes a bowing proposed by the computer, the player can add a constraint (for example, by specifying a bow direction for a particular note), and the algorithm should then recalculate the bowing subject to that constraint.

We have developed an algorithm that calculates a bowing model, given some constraints on particular notes. The algorithm embodies a variety of techniques that string players know. These heuristics are described in the next section, and the following section discusses the software.

4. HEURISTICS FOR CHOOSING A BOWING

Stressed notes tend to be played using a downbow, since it is easier to pull down than up, and also because the frog of the bow is heavier than the tip. For example, in 4/4 time, the first and third beats are played with a downbow, pickup notes are played with an upbow, and the first and last notes of the piece are usually played with a downbow.

Many passages contain a mixture of long and short notes. Naturally, a longer note generally takes more bow than a shorter one. Consequently, it is possible that after taking a long bow, the player may be positioned awkwardly for the next note, and the bowing needs to provide a solution. Furthermore, the common notation does not indicate where the bow will be, so it does not indicate where such problems will occur.

For example, suppose that the player is starting to play a minim (on a downbow), three semi-quavers and then another minim. The player plays the minim and ends up at the tip of the bow. After playing the three semi-quavers at the tip, starting with an upbow, the second minim starts with a downbow. However, the bow is already at the tip, and so the player runs out of bow. Standard bowing notation does not warn a player of this problem, and the player is likely to be taken by surprise while sightreading, or playing a piece that is not familiar. In this situation, one solution is to play two notes on the same bow stroke; this is notated by introducing a slur connecting the first minim and semi-quaver, and causes the second minim to occur on an upbow.

Examples like these show that bowings have constraints. If the player provides a bowing for a passage in the middle of a line of music without considering the requirements of the lines before and after, the bowing will often be wrong (especially if there are no rests delimiting these sections of music).

Constraints require calculation. Players sometimes can be seen to mime playing their instrument while staring at the music—this is because they are calculating the context for a passage by 'playing' the section before it. This calculation can also be done by computer.

Satisfying constraints can be done by standard techniques (but may require clever and unconventional solutions). For example, the player may have to move to a part of the bow while playing notes that are equal in duration. One common way to do this is to 'save' bow, by using strokes that are longer in one direction than the other. If the player is at the tip of the bow and needs to end up at the middle, then the upbows must be longer than the downbows for a few notes. Or suppose the player must remain in one part of the bow (the frog or the tip) because there is a long note coming up that will take the whole bow. The player must then be aware that whatever the durations of the notes in the given passage, the strokes used to play them must be short, so that they do not cause movement away from that part of the bow.

5. RESEARCH CONTEXT

Spohr (19th cent.) and Flesch (early 20th cent.) each wrote about significant bowing problems. In a modern context, the need for a study of bowing has been mentioned as a missing piece of research by a project led by Diana Young [6], where she states that 'A thorough study of different bow strokes and musical performances should follow.' However, while there has been some research on new kinds of bows instrumented with accelerometers and other sensors [4], and also some work on gestures made by the bow arm in playing [3] and a database of bowstrokes [7], calculating bowings from a music text appears to be a new line of research. It is interesting to note that annotations made during orchestral rehearsals have been found to be a valuable resource [5],

suggesting that musical and bowing context is not always as easily understood as string playing tradition would suggest.

6. THE ALGORITHM

We have implemented an algorithm [2] that calculates a bowing model for a piece, using the heuristics given earlier. This algorithm keeps track of the bow direction, and it also has a quantitative representation of the point on the bow hair that contacts the string at the beginning and end of each note. This detailed information allows the algorithm to determine where a bow stroke can be shortened and lengthened to get around problem areas, according to the heuristics.

The algorithm partitions the music into two kinds of blocks: a *simple block* is composed of notes with the same duration, while a *complex block* contains only notes of mixed durations or very long notes.

For simple blocks, the algorithm exploits the fact that the player can make adjustments in the location of the bow. It makes the upbows longer, if the bow position is too close to the tip, and the downbows longer, if the bow position is too close to the frog. This means that the algorithm can work its way out of awkward positions and move back to the middle of the bow, just as the player would.

Complex blocks require some care because they are difficult to bow properly. The algorithm originally handled such a block by backtracking (trying all possibilities). If the first pass failed, then the note being bowed was made shorter, and if that failed, then it was made longer. If all of these failed, then the algorithm backed up to the previous note and tried a new alternative. Backtracking was effective at improving the end of a complex block, but the solutions found were not particularly good, and the task of assessing the value of a solution was incompatible with a backtracking algorithm.

We then implemented an improved algorithm, which calculates all possible bowings through a complex block, and then applies several functions which assess the value of the bowing from a variety of angles. The three different scoring functions, **Average**, **Successive**, and **Extremes**, produce a value from 1 to 100, where 100 is the worst and 1 the best, and the composite score assigned a bowing is the average scores of the three.

Any path which runs out of bow is given 100 without further assessment. The first function, **Average**, takes the average bow position, returning a good score if the average is close to the middle of the bow, and a worse score if it is closer to the tip or frog. While a useful indicator of the value of a bowing, the average may be misleading if there is a sequence of notes at one end followed by another sequence at the other end. For this reason, the second algorithm checks for successive strokes closes to either the frog or the tip. If there are more than 4 strokes at the tip, or more than 2 at the frog, then a worse score is returned. Finally, bowings that require the player to be at an end of the bow make it more likely that the algorithm will run out of bow somewhere along the line, so the third algorithm assigns these worse scores.

7. EXPERIMENTAL RESULTS

We applied the algorithm to six pieces: the first, second and third movements of the Bach double violin concerto in D minor (solo violin 1), and three movements from Handel's sonatas for violin and figured bass (the first movement of the sonata in A major, the third movement of the sonata in D major, and the fourth movement of Handel's sonata in E major). All were assigned viable bowings; none of the solutions run out of bow at some point during the analysis.

Each of these movements had some interesting problems for our algorithm to address. The algorithm assigned bowing annotations without stopping for direction by the musician (unlike our previous work). The musician provided music that was marked with dynamics and wrote in any slurs required for known technical reasons (e. g. the player would run out of bow here otherwise) or musical reasons.

bar	old	new	user	factor	dur
10.44	0.49	0.54	Mb	1.00	0.25
10.50	0.54	0.17	Tb	0.67	2.50
11.13	0.17	0.28	Tb	1.00	0.25
11.19	0.28	0.22	Tb	1.00	0.25
11.25	0.22	0.33	Mb	1.00	0.25
11.31	0.33	0.27	Tb	1.00	0.25
11.38	0.27	0.38	Mb	1.00	0.25
11.44	0.38	0.32	Mb	1.00	0.25
11.50	0.32	0.69	Mb	0.67	2.50

Figure 1. This edited output from the program shows the bar numbers (fractions indicate position of a note within the bar), the old bow position that was current for the previous note, the new bow position, the information provided to the musician (Tb means upper part of the bow, Mb means middle of the bow), and the factor used by the analysis of a complex block to control the length of the stroke(.67 means shorten the stroke, while 1.00 means leave it as it comes). The final column is the note duration as a fraction of the value receiving one beat (so .25 is a sixteenth note (semiquaver) because the time signature is 4 4).

The first movement of the Bach double violin concerto was relatively straightforward (Figure 2). It has lots of blocks of sixteenth or eighth notes (semiquavers or quavers) which allowed the algorithm to maintain the desired average bow position (the middle of the bow). In our example, a note appeared which was so long that it had to be handled in one of two ways. Either the bow had to be at the frog when it started, or the stroke had to be shortened. The algorithm

shortened the duration of that stroke. The notes between the two half notes (minims) were few enough that the second half note was reached before the algorithm moved back to the middle of the bow.

The second movement of the Bach concerto was interesting because it contained a passage which could not be played without shortening the long notes.

The first movement of the first Handel sonata in A major contained lots of tricky sections, where sequences of four or five strokes were twice separated by longer notes that moved in the same direction. This was handled well by the algorithm, but might have been more of a problem if there had been a longer sequence like this.

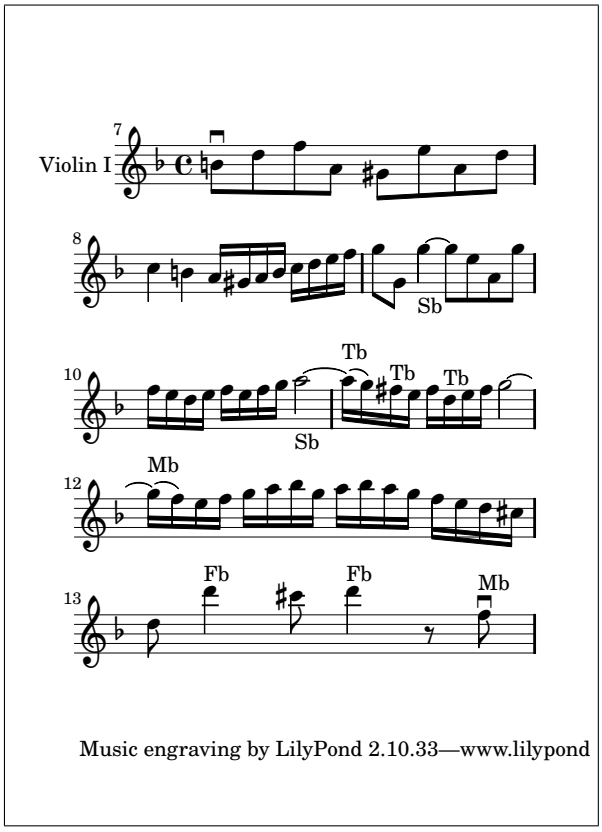

Figure 2. Part of the edited first solo violin part of the Bach double violin concerto in D minor [1], annotated with the results of the algorithm. The annotation Sb means that the stroke should be shortened, Fb means the lower part of the bow, Mb the middle, and Tb the upper part of the bow. Notice that on the last line, the annotations indicate that the player may be getting close to the frog.

The third movement of the Handel sonata in D major was challenging because the tempo was so slow that long notes tended to require a whole bow, leaving little margin for adaptation. The algorithm shortened these.

The fourth movement of the Handel sonata in E major was straightforward, even for uneven sequences.

The third movement of the Bach double violin concerto had one tricky type of passage, in which two sets of four sixteenth notes are played, the last three in each slurred together, and then two slurred triplets. The algorithm marked most strokes in the last bar with an F because they occurred in a mixture of simple and complex blocks which forced the algorithm to have a local and not a global view of the situation.

8. CONCLUSIONS

Professional musicians might find this work interesting, but they are already capable of finding good bowings. Students are the most likely group to benefit from algorithms like this. We also are interested in applying this work to other problems such as building editors for string players where the user can query the editor for information about awkward bowings that may be implied by unedited, or carelessly edited music. One application of the algorithm presented here would be as a checker that determines when an edition has failed to identify a problem that would cause an unwary musician to run out of bow, as in the unedited version of the first solo violin part of the Bach double violin concerto in D minor. Eventually, even orchestral players might benefit from inclusion of such algorithms in digital music stands.

9. REFERENCES

[1] J. S. Bach, *Konzert in d-Moll fur zwei Violinen, Streicher und Basso continuo*. Barenreiter - Verlag Kassel und Deutscher Verlag fur Musik Leipzig, 1986.

[2] C. Hall and B. Findlay, "Simulation of bowing decisions on a string instrument," in *Journees d'Informatique Musicale, Lyon, France*, 1997.

[3] N. Rasamimanana, E. Flety, and F. Bevilacqua, "Gesture analysis of violin bow strokes," in *6th International Gesture Workshop*, 2005, pp. 145–155.

[4] S. Serafin and D. Young, "Bowed string physical model validation through use of a bow controller and examination of bow strokes," in *Proceedings of the Stockholm Music Acoustics Conference, August 6-9*, 2003.

[5] M. Winget, "Heroic frogs save the bow: Performing musician's annotation and interaction behavior with written music," in *7th International Conference on Music Information Retrieval*, 2006.

[6] D. Young, "Wireless sensor system for measurement of violin bowing parameters," in *Proceedings of the Stockholm Music Acoustics Conference, August 6-9*, 2003.

[7] D. Young and A. Deshmane, "Bowstroke database: A web-accessible archive of violin bowing data," in *NIME*, 2007.