Improved Score Following for Acoustic Performances

Bryan Pardo, William Birmingham

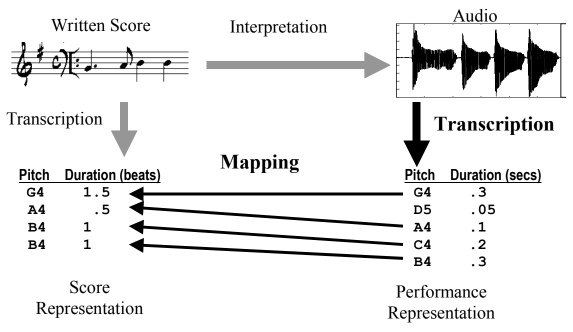
Department of Electrical Engineering and Computer Science, University of Michigan
email: bryanp@umich.edu

Abstract

This paper examines the standard string-matching approach to automated score following of an acoustic performance, looking at limitations imposed by both the score representation used and the matching approach. The approach is then extended through the incorporation of duration information and match-scores based on a model of transcriber error. The resulting score follower promises to allow computer-based accompaniment to follow acoustic performances in situations where earlier systems would fail.

1 Introduction

Automated musical accompaniment that reacts naturally to the human performer is a long-standing goal of a number of computer-music researchers. The ideal is to create a peer musician that can be integrated into an ensemble of human players with minimal need for the humans to adjust their interaction style to accommodate the computer performer. Algorithms that match a score to a human performance are essential for an automated accompanist that reacts appropriately during performance. Systems that perform this function are called score followers. Figure 1 outlines the basic steps in score following. Black arrows illustrate steps commonly performed by score followers. Gray arrows show steps performed by outside agents.



Pitch	Duration (beats)
G4	1.5
A4	.5
B4	1
B4	1

Pitch	Duration (secs)
G4	.3
D5	.05
A4	.1
C4	.2
B4	.3

Figure 1. Basic Steps in Score Following

Score-following research, with the exception of recent work by Grubb and Dannenberg (1997) has concentrated on using string-matching techniques to solve the problem. Unfortunately, such techniques, as currently applied, do not take into account either timing information (rhythm) or transcription error. This paper improves the standard string-matching approach through the incorporation of duration information and explicit modeling of transcriber error.

2 String-match Score Following

A number of researchers (Dannenberg 1984; Dannenberg and Mont-Reynaud 1987; Dannenberg and Mukanio 1988; Puckette and Lippe 1992; Large 1993; Puckette 1995; Desain, Honing et al. 1997; Pardo and Birmingham 2001) have built score followers that assume a score representation where the score is represented as an untimed sequence of pitches from a finite alphabet, such as the keys on a piano, or MIDI pitch numbers. The lower left portion of Figure 1 contains an example (omitting the duration information). They further assume a transcription of the performance using a similar alphabet of pitches, such as is shown in the lower right-hand corner of the figure. Given this, the problem is that of finding the best alignment between two sequences of symbols from the same alphabet.

String matchers find the best alignment between string A and string B by finding the lowest cost transformation of A into B in terms of operations (insertion or deletion of characters). Such matchers are all based on similar techniques and are examples of the “classic” score following approach.

Dynamic-programming based implementations that search for a good alignment of two strings have been used for over 30 years to align gene sequences based on a common ancestor (Needleman and Wunsch 1970), and have also been used in score following by such researchers as by researchers as Dannenberg and Puckette. We now describe a canonical matcher of this kind.

Construct a matrix *AlignScore*, where *AlignScore*(i,j) is the score of the best alignment between the initial segment a_1 through a_i of A and the initial segment b_1 through b_j of B.

The process is initialized by setting *AlignScore*(0,0) = 0. Thereafter, the elements of the matrix are filled in using the following equation.

$$AlignScore(i,j) = \max\begin{cases} AlignScore(i-1,j-1) & +matchScore(a_i,b_j) \\ AlignScore(i-1,j) & -skipPenalty(b_j) \\ AlignScore(i,j-1) & -skipPenalty(a_i) \end{cases}$$

The top line in this equation gives the reward assigned in the score matrix for calling $(a_i\ ,\ b_j)$ a match. The middle line calculates the penalty for skipping b_j and the lowest finds the penalty for skipping a_i.

Define the *matchScore* as follows.

$$matchScore(\alpha,\beta) = \begin{cases} 2, \text{ if } \alpha = \beta \\ -2, \text{ otherwise} \end{cases}$$

The penalty for skipping an element of either sequence is given by

$$skipPenalty(\alpha) = 1$$

As *AlignScore* is filled in, another table must be kept, keeping track of the parent used to fill in each cell (*i,j*). In many cases, it may be that more than one of the parents of cell (*i,j*) gives the maximum value. In this case, all parents may be noted (although, in practice, researchers have noted only a single parent). Once this is done, the best-scoring alignment(s) may be read by starting at the final cell in the matrix and tracing backwards through the series of parents used to generate the score.

For real-time score following, the score sequence is known in advance. This means the number of columns in the table is fixed. Since each cell in the array is filled in using the values of three previously filled in cells, each row may be filled in using a fixed number of steps. The time to fill in the table is linear with respect to the length of the performance, so this method presents no theoretical limit on the ability of a system follow a score in real-time.

Table 1. Alignment Matrix

Score / **Perform.**		**G**	**A**	**B**	**B**
	0	-1	-2	-3	-4
G	-1	**2**	1	0	-1
D	-2	**1**	0	-1	-2
A	-3	0	**3**	**2**	-1
C	-4	-1	**2**	**1**	0
B	-5	-2	1	**4**	**3**

The *AlignScore* matrix for the performance and score from Figure 1 is shown in Table 1. Score elements identify columns. Performance elements identify rows. Arrows show the parent(s) of each element. In this table, a vertical arrow indicates a skip of an element in the performance sequence, a horizontal arrow is a skip of a score element, and a diagonal arrow indicates a match between the two. Cells along the maximal-scoring alignment path(s) are shown in boldface with arrows marking the path. The table shows the following four maximal-scoring alignments between performance and score.

```
SCORE: G-A-BB   G-A-BB   G-ABB   G-AB-B
PERF:  GDACB-   GDAC-B   GDACB   GDA-CB
```

Here, matches are aligned vertically. A skip of a performance element is indicated by a – in the score. A skip of a score element is indicated by a – in the performance.

Of the four alignments, the third is interesting because it shows the matcher aligns a C in the performance to a B in the score. This is a result of the interplay between the skip penalty and the score for a bad match. Varying the *scoreMatch, skipPenalty* functions and the order in which the array is filled allows this simple matcher to emulate the function of both the Dannenberg (84) and Puckette (92) matchers.

3 Modeling Transcription Error

The score matcher described in the previous section has a simplistic *matchScore* function, giving two points if a_i is an exact match of b_j and subtracting two if they are at all different. While this may be adequate, in many cases, for comparing a MIDI sequence to a score, it is too simple a model when comparing a note sequence transcribed from audio using a pitch tracker. Octave displacement, tracking a strong partial, and half step errors due to pitch quantization are all common occurrences. Such errors can be handled gracefully if a probability distribution over the set of possible observations (the note reported by a pitch tracker), given a state (the note played by the performer), is maintained. It would be good to have a *matchScore* function that could take this kind of error into account.

The dynamic-programming algorithm introduced by Gotoh (1982) as described in Durbin, Eddy et al. (1998) finds an optimal global alignment between two sequences, *A* and *B*, taking into account how likely it is that any given element in sequence *A* is related to sequence *B*. We adapt Gotoh's approach to the problem of modeling transcription error.

Here is a summary of our notation. Let there be two distinct, but identical alphabets, A_{score} and A_{trans}. They might contain, for example, the 88 pitches available on a piano.

Assume every element, *e*, of each alphabet occurs independently with some known prior probability, *P*(*e*). For A_{score}, $P(e_{score})$ may be estimated by the proportion of times e_{score} occurs in a representative corpus of scores. This is useful domain knowledge. For example, music for a baritone singer will not have pitches three octaves above middle C. For A_{trans},

$P(e_{trans})$ for each e_{trans} may be estimated from a representative corpus of transcriptions. Given this, the prior probability of a *random co-occurrence* of e_{score} and e_{trans} is given by

$$P(e_{score}, e_{trans} \mid random) = P(e_{score})P(e_{trans})$$

Of course, it may be that e_{score} and e_{trans} occur together because the performer read pitch e_{score} in the score, played it, and the transcriber returned e_{trans} as output. In this case, we call e_{score} and e_{trans} a match and the *match probability* for them may be estimated as follows.

A representative corpus of scores is performed on the appropriate instrument and recorded. These recordings are then hand-transcribed. The hand transcriptions are presumed to report the correct pitch. The output of the pitch tracker may then be compared to the hand transcription. From this, one finds the number of times the pitch tracker labeled a segment as pitch *j*, while the human labeled it as pitch *i*. The human is presumed always correct. Letting *i* be e_{score} and *j* be e_{trans}, this gives us an estimator for the *match probability* of e_{score} and e_{trans} as follows.

$$P(e_{score}, e_{trans} \mid match) = P(e_{score} \mid e_{trans})$$

The *match score* for e_{score} and e_{trans} is the match probability divided by the probability of random co-occurrence, and then taking the log of the resulting value.

$$matchScore(e_{score}, e_{trans}) = \log\left(\frac{P(e_{score}, e_{trans} \mid match)}{P(e_{score}, e_{trans} \mid random)}\right)$$

One nice feature of the log ratio is that when the probability of a match is below that of a random co-occurrence, the value is negative. Similarly, the value is positive when a match is more likely than random chance. This *matchScore* function is applied to the score matcher described in the previous section, replacing the *matchScore* function in that section.

In practice, the values for *matchScore* are pre-computed and stored in an array where the element *i,j* represents the match score for *i*th note in the score alphabet to the *j*th note in the transcription alphabet.

4 Duration Based Skip Penalties

The previous section outlined how to improve classic score matching through explicit modeling of transcription error. Another improvement is the use of timing information in both the score and the transcription. A simple change to the *skipPenalty* takes the relative durations of pitches into account and greatly reduces the number of maximal-scoring paths.

$$skipPenalty(a_i) = k\frac{length(a_i)}{\sum_{j=1}^{|A|} length(a_j)}$$

Here, the *skipPenalty* is now a function of the length of a note (however length is defined) divided by the total length of the performance (or score). In this formula, *k* is a constant that can be used to adjust the relative penalty involved with skipping an element in the sequence. There is no requirement that *k* be the same for both transcription and score and it may be that there should be a much greater penalty for skipping one rather than the other.

5 An Example

To illustrate the utility of the improvements to string-match based score following described in this paper, consider the example of following a performance of *Alouette* performed on alto saxophone.

In the interest of grounding this example in commonly available current technology, all transcription was performed in a quiet office environment using *MicNotator*, a transcription program included with the music notation program *Finale* 2002. All recordings used a Sure SM57 dynamic microphone connected directly to a Soundblaster Platinum card on a Windows PC.

5.1 Calculating Match Scores

In order to create the *matchScore* table, we calculated $P(e_{score}, e_{trans} \mid match)$ for an alto saxophonist over two octaves, starting at the lowest C on the instrument (concert E flat). The saxophonist played a variety of chromatic scales and chord arpeggios for a total of 29 instances of each pitch in the range. Notes were connected, separated, tongued and un-tongued in roughly equal proportions and played at tempos from 30 to 240 notes per minute. Vibrato was discouraged, although some did occur. The resulting count of associations between performed pitch and transcribed pitch was used to estimate both $P(e_{score}, e_{trans} \mid match)$ for all combinations of transcription and score pitches and also, the prior probability of occurrence, $P(e_{trans})$, for all transcription pitches.

Since the score was known before-hand to be *Alouette*, $P(e_{score}) = 0$ for any e_{score} not in *Alouette*. This causes problems for the calculation of the *matchScore*, since the divisor would be 0. We solve this by imposing a minimal value of $P(e_{score}) = 0.01$ and re-normalizing probabilities accordingly. Figure 2 contains the resulting *matchScore* matrix. Here, darker squares indicate higher *matchScore* values.

Were the *matchScore* function from Section 2 illustrated in a similar manner, the only darkened squares would be along the diagonal. This matrix, however, shows a number of dark squares off the diagonal. These are places where the transcriber is likely to make an error. The most common error is that of transcribing certain pitches off by a half-step. Such predictable transcriber errors are given nearly as much weight as correct transcriptions and allow the system to recover from imperfect transcriptions whose errors are similar to ones made when generating the *matchScore* table.

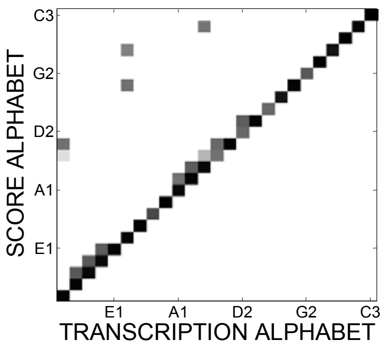


Figure 2. *matchScore* values

5.3 The Alignment

Figure 3 shows a score for *Alouette* (lower staff) and a transcription of a saxophone performance of the score (upper staff) as aligned by a string-matcher based on the techniques described in this paper. The performance did not contain any of the embellishments shown in the transcription. These are entirely due to transcriber error. Further, the assignment of the transcription to written notation in measures is entirely for ease of viewing. The score and transcription representations used by the software are those shown in the lower portion of Figure 1.

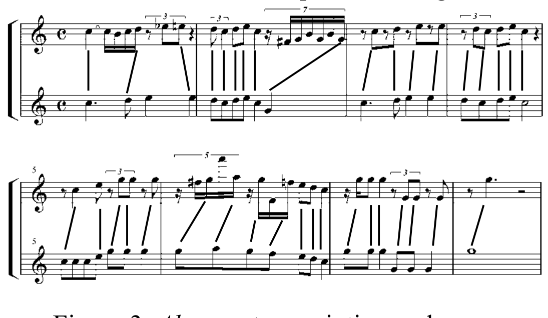


Figure 3. *Alouette* transcription and score

We stated previously that string-based matchers may return a number of equally high-scoring alignments, depending on scoring functions. *Alouette* is an example of this. A matcher using the fixed *matchScore* and *skipPenalty* functions from Section 2 is forced to make an arbitrary choice between equally high-scoring alignments nine times in the course of following this passage. The addition of a duration-based *skipPenalty* and a *matchScore* function based on transcriber error reduces the number of choices to three. A policy of choosing the latest-occuring match for each score note narrows the field to the transcription alignment shown in the figure.

6 Conclusions

We have described a method of incorporating a model of transcriber error and timing information in the standard string-matching approach to score following. Our initial results show this is a good approach that can significantly improve the performance of a score follower based on string-matching techniques. Such a score follower should allow a more natural interaction with the human performer than has been possible with earlier techniques.

7 Acknowledgments

We gratefully acknowledge the support of the National Science Foundation under grant IIS-0085945, and The University of Michigan College of Engineering seed grant to the MusEn project. The opinions in this paper are solely those of the authors and do not necessarily reflect the opinions of the funding agencies.

References

Dannenberg, R. (1984). An On-Line Algorithm for Real-Time Accompaniment. International Computer Music Conference, International Computer Music Association.

Dannenberg, R. and B. Mont-Reynaud (1987). Following an Improvisation in Real Time. International Computer Music Conference.

Dannenberg, R. B. and H. Mukanio (1988). New Techniques for Enhanced Quality of Computer Accompaniment. Proceedings of the International Computer Music Conference, Ann Arbor, MI.

Desain, P., H. Honing, et al. (1997). Robust Score-Performance Matching: Taking Advantage of Structural Information. International Computer Music Conference.

Durbin, R., S. Eddy, et al. (1998). Biological Sequence Analysis, Probabilistic models of proteins and nucleic acids. Cambridge, U.K., Cambridge University Press.

Gotoh, O. (1982). "An improved algorithm for matching biological sequences." Journal of Molecular Biology 162: 705-708.

Grubb, L. and R. Dannenberg (1997). A Stochastic Method of Tracking a Vocal Performer. International Computer Music Conference.

Large, E. W. (1993). "Dynamic Programming for the Analysis of Serial Behaviors." Behavior Research Methods, Instruments and Computers 25(2): 238-241.

Needleman, S. B. and C. D. Wunsch (1970). "A general method applicable to the search for similarities in the amino acid sequence of two proteins." Journal of Molecular Biology 48: 443-453.

Pardo, B. and W. P. Birmingham (2001). Following a musical performance from a partially specified score. Multimedia Technology Applications Conference, Irvine, CA.

Puckette, M. (1995). Score following using the sung voice. International Computer Music Conference.

Puckette, M. and C. Lippe (1992). Score Following In Practice. International Computer Music Conference, International Computer Music Association.