Geometric Algorithms for Melodic Similarity

Mika Laitinen and Kjell Lemström
C-BRAHMS Group, Department of Computer Science
P.O.Box 68 (Gustaf Hällströmin katu 2b)
FIN-00014 University of Helsinki, FINLAND
{mikalait,klemstro}@cs.helsinki.fi

Keywords: MIREX'10, Melodic Similarity, Geometric Matching

1 Introduction

This abstract gives an overview on two algorithms, developed by C-BRAHMS group, submitted to take part in MIREX'10 melodic similarity contest.

Both algorithms work with point-set representation of music and are geometric algorithms designed to process polyphonic music, but work as well with monophonic music. The geometric approach allows us to remove and add notes to both the query and the target, which enables us to find partial matches. The similarity between the query pattern and the target music is defined by the number of elements in $P_M = p_{\pi_0}, \ldots, p_{\pi_x}$ so that P_M matches some $T_M = t_{\tau_0}, \ldots, t_{\tau_x}$ if we apply some invariant. Here $0 \leq \pi_i < m$, $\pi_i < \pi_{i+1}$, $0 \leq \tau_i < n$, $\tau_i < \tau_{i+1}$, m and n denote the number of notes in the query and the target music, respectively. Notes in query and target are denoted with p_i and t_i.

The first algorithm finds matches that satisfy a time-scaling invariant. Essentially this means that the algorithm is able to find intervals that have been arbitrarily enlarged or compressed in time, as long as all the note starting positions have been scaled with the same ratio. Partial matches are allowed, so the algorithm aims to find the largest subsequence of the query pattern that can be matched to the target music under time-scaling invariant.

Similarly, our second algorithm finds matches satisfying a time-warping invariant. Time-warping invariant considers two pieces of music similar if the pitch differences are the same between each element in pattern and target, more formally $(p_{\pi_{i+1}} - p_{\pi_i}).pitch = (t_{\tau_{i+1}} - t_{\tau_i}).pitch$ has to hold. Partial matches are allowed as in the first algorithm.

Since both algorithms are prone to find many false positive matches, we apply windowing, which means that between the selected notes of a match there cannot be too many unselected notes in either the pattern or the target. Simply this means that if we denote the window size with w, then $\pi_i < \pi_{i+1} \leq \min(m-1, i+w)$ and $\tau_i < \tau_{i+1} \leq \min(n-1, i+w)$. This approach both speeds up the algorithms and also makes the found matches more relevant.

Both algorithms are sweep line algorithms, and are computationally efficient in nature. The time complexity is $O(nmw^2 \log n)$ for both algorithms. This means that the algorithms are very efficient with conservatively sized windows.

2 S2: Time-Scale Invariant Algorithm

The idea of this algorithm is to find all maximal subsequences from the target music that match some subsequence in the query pattern. As we are interested in time-scale invariant matches, we aim to find a maximal pattern $p_{\pi_0} \ldots p_{\pi_x}$ so that there exists some σ that for some $t_{\tau_0} \ldots t_{\tau_x}$, $\sigma(p_{\pi_{i+1}} - p_{\pi_i}).time = (t_{\tau_{i+1}} - t_{\tau_i}).time$ holds for all i.

We achieve this by implementing a sweep line algorithm that at each point of the algorithm keeps track of the maximal matches that can be extended at this point. When a match is extended, it is passed forward to the next point x where it can be extended, and when the algorithm reaches this point x, the same procedure is applied. For more detailful technical details, refer Lemström (2010).

For the purpose of the symbolic melodic similarity contest, there are some shortcomings in this algorithm. The algorithm produces quite many false positives, especially when the match is not going to be very long. This happens even with windowing, though windowing reduces this effect. Additionally, when the matches are short, it is possible to find time scaled short melodies that match, but are not relevant in any way. These shortcomings are a tradeoff, since the algorithms are designed to work in a polyphonic environment.

3 W2: Time-Warp Invariant Algorithm

Our time-warp invariant algorithm is very similar in nature to the time-scale invariant algorithm. Instead of finding time scaled intervals, this algorithm allows some additional jittering between the notes, so contrary to the previous algorithm, we now only require that there exists some σ_i for all i so that $\sigma_i(p_{\pi_{i+1}} - p_{\pi_i}).time = (t_{\tau_{i+1}} - t_{\tau_i}).time$. Intuitively this means that we do not require the piecewise vectors to have the same scaling ratio in time.

The actual algorithm works analogously to its time-scaled relative. For further technical details, we refer to Laitinen and Lemström (2010).

As this algorithm has also been designed with polyphonic cases in mind, the previously listed shortcomings apply for this algorithm, too. As time-warp invariant is more flexible, it also allows more false positives, which makes the windowing really significant filtering method here.

References

K. Lemström Towards more robust geometric content-based music retrieval. In *Proc. ISMIR'10, Utrecht, August 2010*

M. Laitinen and K. Lemström Time-warp invariance in geometric music retrieval. Unpublished, 2010