Computing a Geometric Measure of the Similarity Between two Melodies

Greg Aloupis* Thomas Fevens† Stefan Langerman‡ Tomomi Matsui§
Antonio Mesa¶ Yurai Nuñez¶ David Rappaport|| Godfried Toussaint*

June 12, 2003

Abstract

Consider two orthogonal closed chains on a cylinder. The chains are monotone with respect to the angle Θ. We wish to rigidly move one chain so that the total area between the two chains is minimized. This is a geometric measure of similarity between two repeating melodies proposed by Ó Maidín. We present an $O(n)$ time algorithm to compute this measure if Θ is not allowed to vary, and an $O(n^2 \log n)$ time algorithm for unrestricted rigid motions on the surface of the cylinder.

1 Introduction

We have all heard numerous melodies, whether they come from commercial jingles, jazz ballads, operatic aria, or any of a variety of different sources. How a human detects similarities in melodies has been studied extensively [11, 7]. There has also been some effort in modeling melodies so that similarities can be detected algorithmically. Some results in this fascinating study of musical perception and computation can be found in a collection edited by Hewlett and Selfridge-Field [6].

Similarity measures for melodies find application in content-based retrieval methods for large music databases such as *query by humming* (QBH) [5, 12] but also in other diverse applications such as helping prove music copyright infringement [3]. Previous work on melodic similarity is based on methods like approximate string-matching algorithms [1, 8], hierarchical correlation functions [9] and two-dimensional augmented suffix trees [2].

Ó Maidín [10] proposed a geometric measure of the distance between two melodies, M_a and M_b. The melodies are modelled as monotonic pitch-duration rectilinear functions of time as depicted in Figure 1. Ó Maidín measures the distance between the two melodies by the area between the two polygonal chains. This rectilinear representation of a melody is equivalent to the triplet melody representation in [9].

In a more general setting such as music retrieval systems, we may consider matching a short query melody against a larger stored melody. Furthermore, the query may be presented in a different *key* (transposed in the vertical direction) and in a different *tempo* (scaled linearly in the horizontal direction). Francu and Nevill-Manning [4] compute the minimum area between two such chains, taken over all possible transpositions. They do this for a constant number of pitch values and scaling factors, and each chain is divided into m and n equal time-steps. They claim (without descibing in detail) that their algorithm takes $O(nm)$ time, where n and m are the number of unit time-steps in each query. This time bound can be achieved with a brute-force approach. In this paper we solve a similar problem, in a more general setting.

In some music domains such as Indian classical music, Balinese gamelan music and African music, the melodies are cyclic, i.e. they repeat over and over. In Indian music these cyclic melodies are called *talas* [13]. Two such monophonic melodies may be represented by orthogonal

*School of Computer Science, McGill University. {athens,godfried}@uni.cs.mcgill.ca

†Department of Computer Science, Concordia University. fevens@cs.concordia.ca

‡Chargé de Recherches du FNRS, Département d'Informatique, Université de Bruxelles. Stefan.Langerman@ulb.ac.be

§Department of Mathematical Informatics, Graduate School of Information Science and Technology, University of Tokyo. tomomi@misojiro.t.u-tokyo.ac.jp

¶Facultad de Matematica y Computacion, Universidad de la Habana. tonymesa@matcom.uh.cu , yurainr@yahoo.com

||School of Computing, Queen's University. daver@cs.queensu.ca

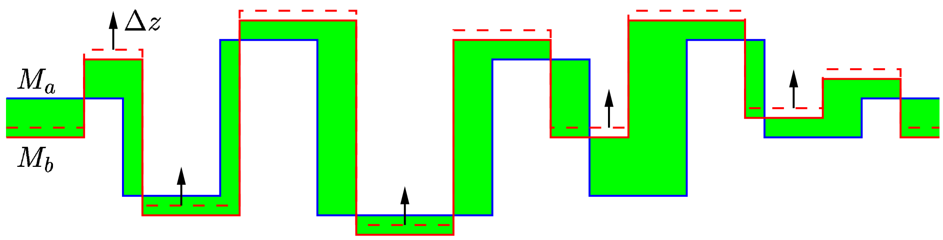

Figure 1: The area between two melodies, M_a and M_b.

polygonal chains on the surface of a cylinder, as shown in Figure 2. This is similar to Thomas Edison's cylinder phonographs, where a melody repeats after the cylinder has rotated 2π degrees. We consider the problem of computing the minimum area between two such chains of size n, over all translations on the surface of the cylinder.

We present two algorithms to find the minimum area between two given orthogonal melodies with periods of 2π. The first algorithm will assume that the Θ direction is fixed. The second algorithm will find the minimum area when both the z and Θ relative positions may be varied. We will assume that the vertices defining M_a and M_b are given in the order in which they appear in the melodies.

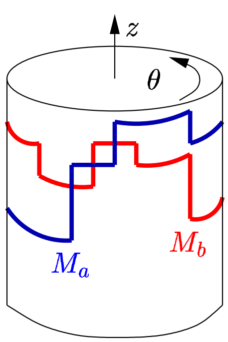

Figure 2: Two orthogonal periodic melodies.

2 Minimization with respect to z direction

In the first algorithm, we will assume that both melodies are fixed in the Θ direction Without loss of generality, we will assume that melody M_a is fixed in both directions, so all motion is relative to M_a. In Figure 1 we show the area between two melodies, and a small shift of M_b in the z direction.

To see how the area between the two melodies changes as M_b moves in the z direction, consider a set of lines defined by all vertical segments of the melodies as shown in Figure 3. This set of lines partitions the area between the melodies into quadrangles C_i, $i = 1, \ldots, k$, defined by two vertical lines and two horizontal segments, one from each melody. Note that k is at most $\frac{n}{2}$. If M_b starts completely below M_a, then as M_b moves in the positive z direction, the lower horizontal segment (from M_b) will approach the upper fixed horizontal segment while the area of C_i decreases linearly until the horizontal segments are coincident (when this occurs, the area of C_i is zero). Then the upper horizontal segment (now from M_b) will move away from the lower fixed horizontal segment while the area of C_i increases linearly. We will consider the vertical position of M_b to be the z-coordinate of its first edge. When this edge overlaps the first edge of M_a, we have $z = 0$. Let $A_i(z)$ denote the area of C_i as a function of z. At z_i, $A_i(z_i) = 0$. These k positions of M_b where an $A_i(z)$ is zero will be called z-events. The slope of $A_i(z)$ is determined by the length of the horizontal segments of C_i. The total area between M_a and M_b is given by $A(z) = \sum_{i=1}^{k} A_i(z)$. Note that since $A(z)$ is the sum of piecewise-linear convex functions, it too is piecewise-linear and convex. Furthermore its minimum must occur at a z-event.

Theorem 1. *A minimum for $A(z)$ can be computed in $O(n)$ time.*

Proof. The function $A(z)$ is given by $A(z) = \sum w_i |z_{bi} - z_{ai}|$, where z_{bi} is the vertical coordinate of M_b in C_i, z_{ai} corresponds to M_a, and w_i is the weight (width) of C_i, as shown in Figure 3. Let α_i denote the vertical offset of each horizontal segment in M_b from z_{b1}. Thus we have $z_{bi} = z_{b1} + \alpha_i$, and $A(z) = \sum w_i |z_{b1} - (z_{ai} - \alpha_i)|$. Finally, notice that the term $z_{ai} - \alpha_i$ is equal to z_i. Thus we obtain $A(z) = \sum w_i |z_i - z_{b1}|$. This is a weighted sum of distances from z_{b1} to all the z-events. The minimum is the weighted univariate median of all z_i and can be found in $O(n)$ time [14]. This median is the vertical coordinate that z_{b1} must have so that $A(z)$ is minimized. Once this is done, it is straightforward to compute the sum of areas in linear time. ∎

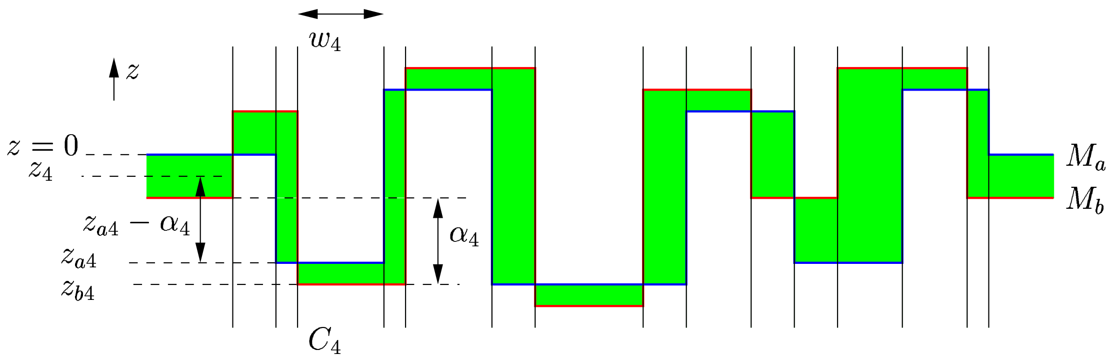

Figure 3: Contribution of C_4 to area calculation.

3 Minimization with respect to z and Θ directions

If no vertical segments among M_a and M_b share the same Θ coordinate, then M_b may be shifted in at least one of the two directions $\pm\Theta$ so that the sum of areas does not increase. This means that in order to find the global minimum, the only Θ coordinates that need to be considered are those where two vertical segments coincide. Thus our first algorithm may be applied $O(n^2)$ times to find the global minimum in a total of $O(n^3)$ time.

We now propose a different approach to improve this time complexity.

As described in the previous section, for a given Θ, the area minimization resembles the computation of a weighted univariate median. When we shift M_b by $\Delta\Theta$, we are essentially changing the input weights to this median. Some C_i grow in width, some become narrower, and some stay the same width. As we keep shifting, at Θ coordinates where vertical segments coincide, we have the destruction of a C_i and creation of another C_i. An important observation is that the rate at which the changing C_i grow or shrink is unique at any given instant.

Let us store the z-events and their weights in the leaves of a balanced binary search tree. Each leaf represents one C_i. The leaves are ordered by the value z_i. Each leaf also has a label to distinguish between C_i that are growing, shrinking, or unaffected when M_b is shifted in the positive Θ direction. At every node with subtree T we store:

- G_T: The number of leaves in T that represent growing C_i.
- $W_T^+ = \sum w_i$: The sum of weights over such leaves.
- S_T: The number of leaves in T that represent shrinking C_i.
- $W_T^- = \sum w_i$: The sum of weights over such leaves.
- $W_T^0 = \sum w_i$: The sum of weights of the remaining leaves in T.

This allows us to calculate the weighted median of all z_i, by traversing the tree from root to leaf, always choosing the path that balances the total weight on both sides of the path. The time for this is $O(\log n)$.

Suppose that we shift M_b by some offset $\Delta\Theta$, which is small enough such that no vertical segments overlap during the shift. Each w_i that contributes to W^+ must be increased by $\Delta\Theta$, and each w_i that contributes to W^- must decrease by this amount. Instead of actually updating all our inputs, we just maintain a global variable representing the total offset in the Θ direction.

The total weight of a subtree T is now given by $W_T^+ + G_T\Delta\Theta + W_T^- - S_T\Delta\Theta + W_T^0$.

When we shift to a point that two vertical segments share the same Θ coordinate, we potentially eliminate some C_i, create a new C_i, or change type of C_i. There is only a constant number of such events at each critical Θ coordinate. The weight given to a created leaf must equal $-\Delta\Theta$. Each of these events involves $O(\log n)$ work to update the information stored in the ancestors of a newly inserted/deleted/altered leaf. There are $O(n^2)$ such instances where this must be done and where the median must be recomputed, so the total time to compute all candidate positions of M_b is $O(n^2 \log n)$.

At every Θ coordinate where we recalculate the median, we also need to calculate the integral of area between the two melodies. For a given median z^*, the area summation for those C_i for which $z_* > z_i$ has the form $\sum w_i(z_* - z_i)$.

This may be calculated in $O(\log n)$ time if we know the value of this summation for every subtree. In order to do this, we store some additional information at every node in our tree. Specifically, if we just consider the contribution to this sum from leaves representing growing C_i, we have $\sum(w_i + \Delta\Theta)(z_* - z_i)$. For a subtree T, this becomes $z_* W_T^+ + z_* \Delta\Theta - \sum w_i z_i - \Delta\Theta \sum z_i$. We see that the additional information that we must store is $\sum w_i z_i$ and $\sum z_i$. To handle shrinking weights, we must also store these two summations taken over the shrinking C_i belonging to T. For unaffected weights, we just need $\sum z_i$. All of these stored values may be updated in $O(\log n)$ time when inserting/deleting leaves in the tree. We must also perform this calculation for all z-events greater than z^*.

Since at every critical Θ position we can calculate the median and integral of area in $O(\log n)$ time, we obtain the following theorem:

Theorem 2. *Given two orthogonal periodic melodies of size n, an relative placement such that the area between the melodies is minimized can be computed in $O(n^2 \log n)$ time.*

4 Remarks

A special case of the general problem is to match two melodies while varying only Θ. This problem seems to be just as difficult as global matching. Extensions may involve more complicated representations such as piecewise-linear pitch, the use of integer weights/heights, or the consideration of scaling.

Acknowledgements

We wish to thank all participants of the Second Cuban Workshop on Algorithms and Data Structures, held at the University of Havana, April 13–19, 2003.

References

[1] David Bainbridge, Craig G. Nevill-Manning, Ian H. Witten, Lloyd A. Smith, and Rodger J. McNab. Towards a digital library of popular music. In *Proceedings of the Fourth ACM International Conference on Digital Libraries*, 1999.

[2] A.L.P. Chen, M. Chang, J. Chen, J. L. Hsu, C. H. Hsu, and S. Y. S. Hua. Query by music segments: An efficient approach for song retrieval. In *Proc. of IEEE International Conference on Multimedia and Expo*, 2000.

[3] C. Cronin. Concepts of melodic similarity in music-copyright infringement suits. In W.B. Hewlett and E. Selfridge-Field, editors, *Melodic Similarity: Concepts, procedures and applications.* MIT Press, Cambridge, Massachusetts, 1998.

[4] Cristian Francu and Craig G. Nevill-Manning. Distance metrics and indexing strategies for a digital library of popular music. In *Proc. IEEE International Conference on Multimedia and EXPO (II)*, 2000.

[5] Asif Ghias, Jonathan Logan, David Chamberlin, and Brian C. Smith. Query by humming: Musical information retrieval in an audio database. In *ACM Multimedia*, pages 231–236, 1995.

[6] W.B. Hewlett and E. Selfridge-Field, editors. *Melodic Similarity: Concepts, procedures and applications.* MIT Press, Cambridge, Massachusetts, 1998.

[7] L. Hofmann-Engl. Melodic similarity - a conceptual framework. In *Proc. 2nd International Conference on Understanding and Creating Music*, Naples, 2002.

[8] K. Lemström. *String Matching Techniques for Music Retrieval.* PhD thesis, University of Helsinki, Faculty of Science, Department of Computer Science, 2000.

[9] L.Lu, H.You, and H.-J. Zhang. A new approach to query by humming in music retrieval. In *ICME2001*, Tokyo, 2001.

[10] D. Ó Maidín. A geometrical algorithm for melodic difference. *Computing in Musicology*, 11:65–72, 1998.

[11] I.C. Martinez. Contextual factors in the perceptual similarity of melodies. *The Online Contemporary Music Journal*, 7, 2001.

[12] Jong-Sik Mo, Chang Ho Han, and Yoo-Sung Kim. A melody-based similarity computation algorithm for musical information. In *1999 Workshop on Knowledge and Data Engineering Exchange*, page 114, 1999.

[13] Robert Morris. Sets, scales, and rhythmic cycles: a classification of *talas* in Indian music. In *21st Annual Meeting of the Society for Music Theory*, Chapel Hill, NC, December 1998.

[14] A. Reiser. A linear selection algorithm for sets of elements with weights. *Information Processing Letters*, 7:159–162, 1978.