Algorithms for Computing Geometric Measures of Melodic Similarity *

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

November 15, 2004

Abstract

Consider two orthogonal closed chains on a cylinder. These chains are monotone with respect to the tangential Θ direction. We wish to rigidly move one chain so that the total area between the two is minimized. This minimization is a geometric measure of similarity between two melodies proposed by Ó Maidín. The Θ direction represents time and the axial direction, z, represents pitch. Let the two chains have n and m vertices respectively, where $n \geq m$, We present an $O(n+m)$ time algorithm if Θ is fixed, and an $O(nm\log(n+m))$ time algorithm for general rigid motions. These bounds also apply for planar orthogonal monotone open chains, where area is measured only within the common domain of the two chains in the direction

*This paper extends the results presented by the authors at CCCG'03 [1]

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

[‡]Department of Computer Science and Software Engineering, Concordia University. fevens@cs.concordia.ca

[§]Chercheur qualifié 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

of monotonicity. We extend the algorithm for which Θ is fixed to non-orthogonal melodies.

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 [16, 11, 19]. 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 [10].

Similarity measures for melodies find application in content-based retrieval methods for large music databases such as *query by humming* (QBH) [8, 17] but also in other diverse applications such as helping prove music copyright infringement [6]. Previous work on rhythmic and melodic similarity is based on methods like one-dimensional edit distance computations [21], approximate string-matching algorithms [3, 12], hierarchical correlation functions [13], two-dimensional augmented suffix trees [4], transportation distances [22, 14], and maximum segment overlap [23].

Ó Maidín [15] proposed a geometric measure of the difference 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. This rectilinear representation of a melody is equivalent to the triplet melody representation in [13]. Ó Maidín measures the difference between the two melodies by the minimum area between the two polygonal chains, allowing vertical translations. The area between two polygonal chains is found by integrating the absolute value of the vertical L_1 distance between M_a and M_b over the domain Θ. Arkin et al. [2] show that the minimum integral of any distance L_p ($p \geq 1$) between two orthogonal cyclic chains, (allowing translations along Θ and z) is a metric.

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 [7] compute the minimum area between two such chains, taken over all possible transpositions. They do this for a constant

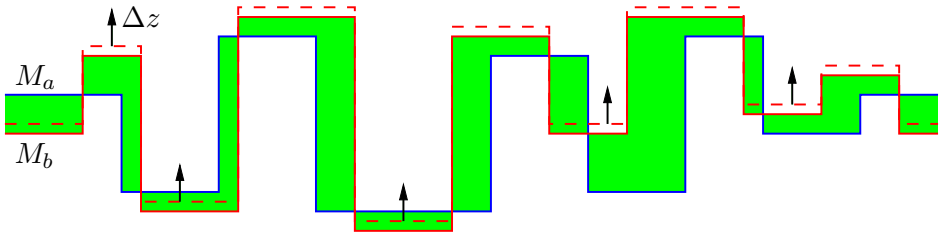

Figure 1: The area between two melodies, M_a and M_b.

number of pitch values and scaling factors, and each chain is divided into m and n equal time-steps. They claim (without describing 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 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* [18]. Two such monophonic melodies may be represented by orthogonal polygonal chains on the surface of a cylinder, as shown in Figure 2. This is similar to Thomas Edison's cylinder phonographs, where music is represented by indentations around the body of a tin foil cylinder. We consider the problem of computing the minimum area between two such chains, over all translations on the surface of the cylinder.

We present two algorithms to find the minimum area between two given orthogonal melodies, M_a and M_b of size n and m respectively ($n > m$). The algorithms may be used for cyclic melodies or in the context of retrieving short patterns from a database (open planar orthogonal chains). We have chosen to describe the algorithms for the case where the melodies are cyclic. The first algorithm, given in section 2, will assume that the Θ direction is fixed. The second algorithm, described in section 3, will find the minimum area when both the z and Θ relative positions may be varied. In each case, we will assume that the vertices defining M_a and M_b are given in the order in which they appear in the melodies. In section 4 we discuss natural extensions, both for the polygonal description of melodies and for the types of queries.

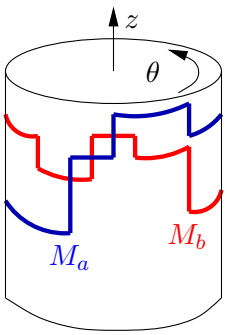

Figure 2: Two orthogonal periodic melodies.

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 motions are 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 rectangles C_i, $i = 1, \ldots, k$, each defined by two vertical lines and two horizontal segments, one from each melody. Note that k is at most $\frac{n+m}{2}$. The area between M_a and M_b is the sum of the areas of all C_i. If M_b starts completely below M_a and moves in the positive z direction, then for any given C_i the lower horizontal segment (from M_b) will approach the upper fixed horizontal segment while the area of C_i decreases linearly. This happens until the horizontal segments are coincident (and 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. We define $z = 0$ to be the position where this edge overlaps the first edge of M_a. Let $A_i(z)$ denote the area of C_i as a function of z. Define z_i to be the coordinate at which $A_i = 0$. These k positions of M_b where some A_i becomes 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 $O(n+m)$ 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 $O(k)$ time [20]. 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 $O(k)$ time. Recall that k is at most $\frac{n+m}{2}$.

□

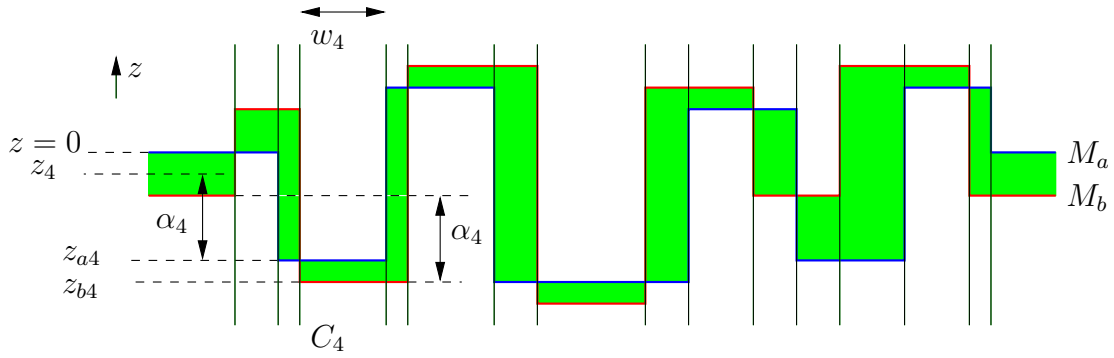

Figure 3: Contribution of C_4 to area calculation.

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 $O(nm)$ times to find the global minimum in a total of $O(nm(n+m))$ 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 all C_i grow (or shrink) at the same rate.

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 infinitesimally in the positive Θ direction. At every node with subtree T we store:

- W_T : The sum of weights of all leaves in T.
- D : The number of growing leaves minus the number of shrinking leaves in T.

The weighted median of all z_i may be calculated 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 $O(\log k)$.

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 belonging to a growing leaf must be increased by $\Delta\Theta$, and each w_i belonging to a shrinking leaf must be decreased by this amount. Instead of actually updating all our inputs, we just maintain a global variable $\Delta\Theta$,representing the total offset in the Θ direction. The total weight of a subtree T is now $W_T + D\Delta\Theta$.

When we shift to a position where 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. The number of such changes is constant for each pair of collinear vertical segments. The weight given to a created leaf must equal $-\Delta\Theta$. Each of these changes involves $O(\log k)$ work to update the information stored in the ancestors of a newly inserted/deleted/altered leaf. There are $O(nm)$

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 $O(nm \log(n+m))$.

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 $O(\log k)$ time if we know the value of this summation for every subtree. In order to do this, we store some additional information at every subtree T. Specifically, the area is given by
$z_*(W_T + D\Delta\Theta) - \sum(w_i z_i) - \Delta\Theta \sum(I z_i)$,
where in the second summation I takes the values $(+1, 0, -1)$ for growing, unchanged and shrinking leaves respectively. These two summations are the additional parameters that need to be stored, and they may be updated in $O(\log k)$ time at every critical Θ coordinate.

We must also perform a similar $O(\log k)$ time calculation of $\sum w_i(z_i - z_*)$, for all $z_i > z_*$. No additional parameters are needed for this.

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

Theorem 2. *Given two orthogonal periodic melodies with n and m vertices, a relative placement such that the area between the melodies is minimized can be computed in $O(nm \log(n+m))$ time.*

The analysis above may be used for the problem of matching two planar orthogonal monotonic open chains. Clearly if we are only interested in varying one direction, an optimal placement may be found in linear time. If the direction of monotonicity is the x-axis, then this problem is more interesting if one of the two chains has a shorter projection onto the x-axis. This "shorter" chain reminds us of a short motif that we might search for in a larger database of music. For this problem, we measure area only within the common domain of the two chains along the x-axis. Naturally, the projection of the shorter chain must be entirely covered by the projection of the longer chain.

Corollary 3. *Given two planar orthogonal chains monotone with respect to the x-axis, with n and m vertices respectively, a relative placement such that the area between the chains is minimized can be computed in $O(nm \log(n+m))$ time.*

Arkin et al. [2] showed that two polygonal shapes may be compared by parametrizing their boundary lengths and examining their orientation differences. They showed that their measure, which is invariant to scaling, rotation and translation, can be computed by finding the minimum integral of the vertical distance between two orthogonal chains, which are constructed in a preprocessing step. In fact some of their techniques are similar to those given in this section. However, they chose to use the L_2 distance (as opposed to the L_1 distance used here), for which the optimal z-position at any θ can be computed in $O(1)$ time. The complexity of their algorithm is dominated by sorting the $O(nm)$ critical θ events. They indicated that their algorithm offers no improvement over a $O(n^3)$ time brute-force approach for the L_1 metric.

4 Extensions

4.1 Higher dimensions

Consider a simple orthogonal open chain which is monotone with respect to the x-axis. Furthermore, at any particular x-coordinate suppose that the chain has at most two edges (in the y- and/or z-directions). This is an extension of the melody representation which we have seen so far. The x-axis still represents time, but perhaps now the other axes might represent pitch, loudness, timbre or chord density. In the plane, the measurement made was an integral of the pitch (height) difference taken over a domain in the x-axis. Here, we still wish to minimize an integral of the distance between two chains over all common x-coordinates. Whether this should be Euclidean distance or perhaps the L_1 distance is debatable. The latter is definitely easier to compute. Suppose that we only allow motions of the chains M_a and M_b in the y- and z-directions. Minimizing the sum of pairwise Euclidean distances is equivalent to the Weber problem, which involves finding a point with minimum sum of distances to points in a given set. It is not possible to find an exact solution to the Weber problem (also known as the generalized Fermat-Torricelli problem; see [9]). Using the L_1 metric, the function to minimize is $\sum w_i(|z_{bi} - z_{ai}| + |y_{bi} - y_{ai}|)$. This may be split into two terms, $\sum w_i|z_{bi} - z_{ai}| + \sum w_i|y_{bi} - y_{ai}|$. Thus we just have to make two univariate median computations to find the optimal (y, z) placement for a particular relative position of the two chains in the x-direction. In $\mathbb{R}^d$ we can accomplish

this task in $O(dn)$ time. The decoupling of the two coordinates allows us to update each median separately at every critical x coordinate. In $\mathbb{R}^3$ there are still $O(nm)$ critical x coordinates and $O(n+m)$ weights/leaves, so the time complexity is the same as for planar chains. If we let n and m be the total number of edges parallel to the x-axis for two chains, then in $\mathbb{R}^d$ the time complexity becomes $O(mnd\log(m+n))$, using $O(dn)$ space. Note that only these edges are significant in any of the computations we have made so far.

4.2 Scaling

Here we consider the effect of scaling planar chains, either in the vertical or horizontal directions.

If we shrink the shorter chain horizontally, the domain of the integral becomes smaller, so the total area will tend to zero eventually. How should we deal with this? It seems reasonable to normalize by computing the total area over the domain of the smaller chain. It is equivalent to fix the shorter chain at unit domain length and modify the larger chain instead. Its domain would expand from unit length to some value where its narrowest strip has unit width.

Let an *x-value* be an x-coordinate where there are vertical segments from both chains.

Lemma 4. *For the scaling method proposed above, the optimal scaling of the larger chain occurs at a position where two or more x-values occur.*

Proof. For a particular scaling value we know that the optimal placement of the larger chain occurs when we have an x-value. This follows from the arguments given in section 2. Suppose that somehow we know the optimal scaling factor. Assume that there is only one x-value and we know which two vertical segments are aligned. Now we can keep scaling the large chain while using the x-value as an "anchor". One of the two scaling directions will improve the area minimization, at least until we obtain another x-value. □

This means that we have $O(n^2m^2)$ candidate configurations, so a brute-force algorithm would take $O(n^3m^2)$ time using $O(n)$ space. The lemma also applies to vertical scaling. In this case a brute-force algorithm would have a time complexity of $O(n^3m^3\log(n+m))$.

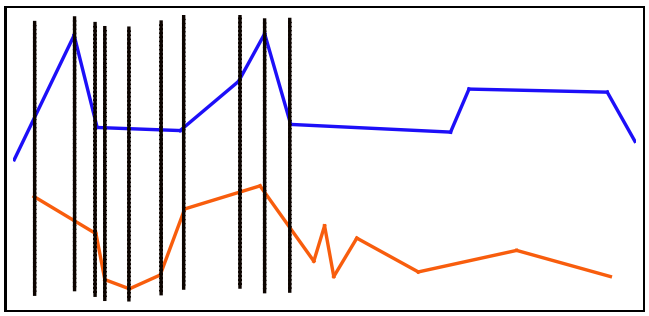

Figure 4: Two monotone chains and their strips.

4.3 Non-orthogonal chains

In the preceding sections it is assumed that a melody may be divided into intervals, and within each interval the pitch (or volume/timbre) remains constant. In a more general setting, these features may vary within each interval. A further step in this direction is to consider monotonic piecewise linear chains. Consider two such planar chains. Let us divide the plane into strips, just as we had for orthogonal chains. In this case, a vertical boundary is placed at every vertex, as shown in Figure 4.

Thus within every strip we have two linear segments. Suppose we vary only the relative pitch of the chains. As one chain is moved down from infinity, within a given strip the area decreases linearly until the two segments touch inside the strip. Then the area decreases quadratically until the midpoints of the segments intersect. Of course, the reverse occurs as we keep moving the chain down. The overall area function of each strip C_i is now a symmetric convex function, which is part linear and part quadratic (around the symmetric point). The total area is a sum of n functions, such as those shown in Figure 5.

The area function is convex and piecewise quadratic with $O(n)$ inflection points. Specifically, in the aggregate function an inflection point will exist only at a coordinate where some individual function changes from linear to quadratic. There are two such points per individual function. Note that the minimum of the aggregate function need not occur at an inflection point, unlike the case of orthogonal chains. Now, it is possible for the minimum to exist between two consecutive inflection points. This would be the only region between two successive inflection points where the function is not

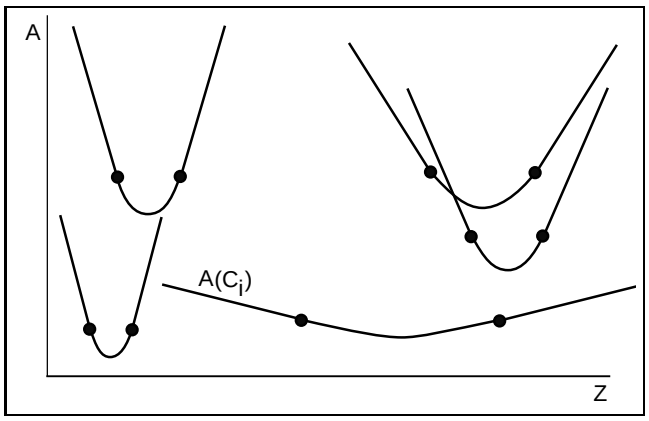

Figure 5: A set of area functions from the C_i strips.

monotone.

To compute the minimum of the aggregate function, we give the following algorithm:

1. Let R be the set of individual area functions. Let F be a single quadratic term, initialized at zero.

2. Compute Q_1, the median of the x-coordinates of the minima of all functions in R, as shown in Figure 6.

3. Compute the value and gradient of the total area function at Q_1, by querying F and all functions in R. If not at the global minimum, assume without loss of generality that the minimum is to the left of Q_1.

4. For the subset of functions in R whose minima are to the right of Q_1, compute the median Q_2 of their *left* inflection points. Q_2 splits the subset into the *left* group and the *right* group.

5. If $Q_2 \geq Q_1$, as shown in Figure 7, replace all functions in the *right* group with a single linear term, which is a summation of all individual left-hand linear terms. Update F by adding this term to it. Remove the *right* group from R.

6. Else if $Q_2 < Q_1$, as shown in Figure 8, compute the gradient of the total function at Q_2. If the global minimum is to the left of Q_2, follow the

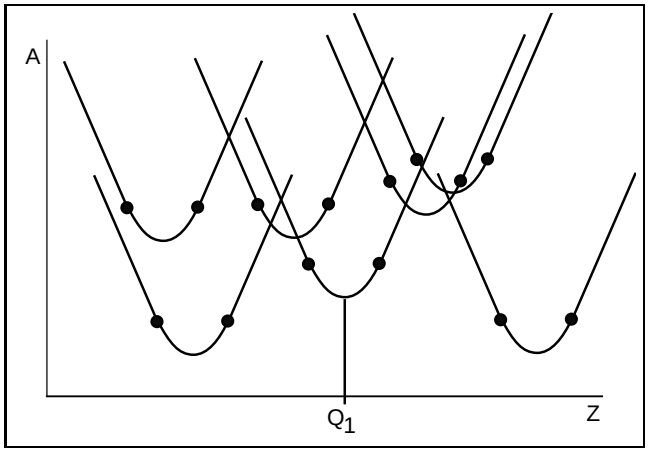

Figure 6: The median Q_1 of function minima.

instructions of step 5 on the *right* group. Otherwise if the minimum is between Q_2 and Q_1, replace all functions in the *left* group with a single quadratic term, which is a summation of all individual quadratic terms. Then update F and remove the *left* group from R.

7. Go to step 2.

The algorithm does $O(|R|)$ work in each iteration, and a constant fraction of R is removed each time. Thus the total time is $O(n)$, by a simple geometric series summation, given in [5].

Theorem 5. *The minimum area between two x-monotone chains, found over all vertical translations, can be computed in $O(n)$ time.*

Updating the aggregate function as we shift one of the chains along the x-axis appears to be non-trivial. It is no longer true that the optimal position must occur when vertices from each chain are aligned vertically. Also, when we make a small shift along the x-axis, not only do the two linear parts of each individual function change slope, but the center of symmetry of each function also may shift (Recall that these are functions of the z-coordinate). These changes depend on the slopes of our chains within each strip and are not difficult to compute on an individual basis. However understanding their aggregate effect is a different matter. To rephrase, each strip now has *three* "z-events" instead of one (the two boundaries between linear and quadratic

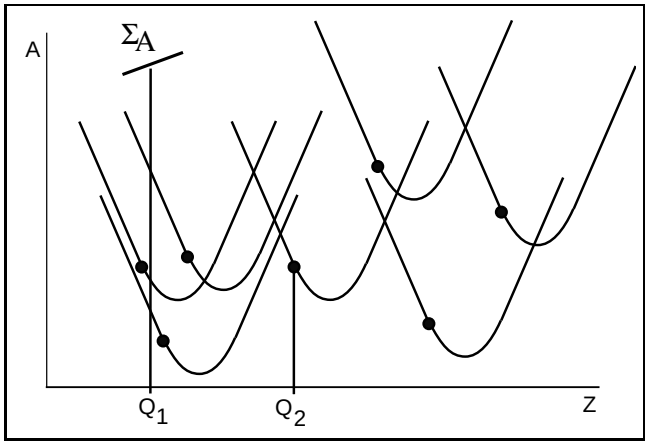

Figure 7: The median Q_2 of left inflection points.

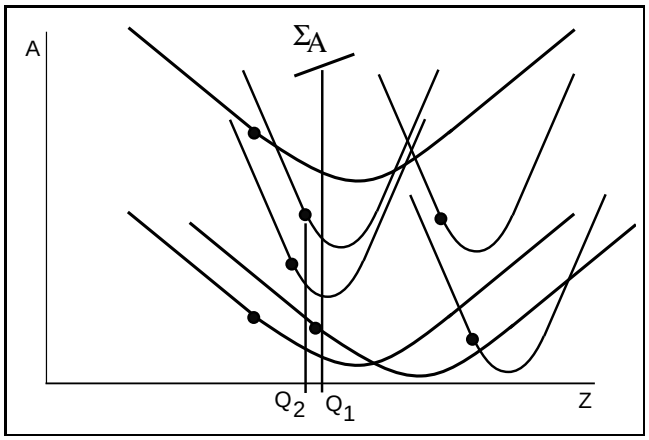

Figure 8: Q_2 to the left of Q_1.

forms, plus the center of symmetry). To make things worse, the z-events change position as a chain is shifted along θ. So if a tree is used to maintain the median, it will be necessary not only to insert/delete leaves but also to rearrange the order of leaves (to say the least).

4.4 Integer weights/heights

Here we discuss the cases where only certain pitches (heights) and/or weights are allowed.

If there are $O(1)$ height differences allowed, we can sort all critical points in $O(nm \log m)$, and sweep aong each height difference horizontally, updating the area function in $O(1)$ time per critical point (i.e. $O(mn)$ per height difference), so the time complexity is dominated by the sorting step. Even in the simplest case, where we just wish to compute the minimum area while keeping z fixed, we do not know how to avoid sorting all critical positions.

If all weights are equal (i.e. we have evenly spaced sampling of melodies), then each median computation takes $O(m)$ time and there are $O(n)$ critical positions. Thus a brute force approach takes $O(nm)$ time. A direct implementation of our tree algorithm would take $O(nm \log m)$ time, since at each of the $O(n)$ critical positions we would have to update all $O(m)$ leaves of our tree. It is possible that this can be greatly improved.

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. We also thank Remco Veltkamp and Rainer Typke for bringing [2] to our attention.

References

[1] Greg Aloupis, Thomas Fevens, Stefan Langerman, Tomomi Matsui, Antonio Mesa, Yurai Nunez, David Rappaport, and Godfried Toussaint. Computing a geometric measure of the similarity between two melodies. In *Proceedings of the 15th Canadian Conference on Computational Geometry*, pages 81–84, Halifax, August 2003.

[2] Esther Arkin, Paul Chew, Daniel Huttenlocher, Klara Kedem, and Joseph Mitchell. An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):209–216, March 1991.

[3] 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.

[4] Arbee L.P. Chen, Maggie Chang, Jesse Chen, Jia-Lien Hsu, Chih-How Hsu, and Spot Y. S. Hua. Query by music segments: An efficient approach for song retrieval. In *Proc. IEEE International Conference on Multimedia and EXPO (II)*, pages 873–876, 2000.

[5] T. Cormen, Ch. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. Carnegie Mellon University, September 2001.

[6] Charles 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.

[7] 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.

[8] 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.

[9] C. Groß and T. Strempel. On generalizations of conics and on a generalization of the Fermat-Torricelli problem. *American Mathematical Monthly*, 105(8):732–743, 1998.

[10] Walter B. Hewlett and Eleanor Selfridge-Field, editors. *Melodic Similarity: Concepts, procedures and applications*. MIT Press, Cambridge, Massachusetts, 1998.

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

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

[13] Lie Lu, Hong You, and Hong-Jiang Zhang. A new approach to query by humming in music retrieval. In *ICME2001*, pages 22–25, Tokyo, 2001.

[14] Anna Lubiw and Luke Tanur. Pattern matching in polyphonic music as a weighted geometric translation problem. In *Proc. 5th International Conference on Music Information Retrieval*, pages 289–296, Barcelona, Spain, October 10-14 2004. Universitat Pompeu Fabra.

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

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

[17] 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.

[18] 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.

[19] Daniel Müllensiefen and Klaus Frieler. Measuring melodic similarity: Human vs. algorithmic judgements. In *Proceedings of the Conference on Interdisciplinary Musicology*, Graz, Austria, April 2004.

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

[21] Godfried T. Toussaint. A comparison of rhythmic similarity measures. In *Proc. 5th International Conference on Music Information Retrieval*, pages 242–245, Barcelona, Spain, October 10-14 2004. Universitat Pompeu Fabra.

[22] Rainer Typke, Panos Giannopoulos, Remco Veltkamp, Frans Wiering, and René van Oostrum. Using transportation distances for measuring melodic similarity. In *Proceedings of the International Conference on Music Information Retrieval ISMIR*, pages 107–114, 2003.

[23] Esko Ukkonen, Kjell Lemström, and Vell Mäkinen. Geometric algorithms for transposition invariant content-based music retrieval. In *Proceedings of the International Conference on Music Information Retrieval ISMIR*, pages 193–199, 2003.