ITERATIVE DEEPENING FOR MELODY ALIGNMENT AND RETRIEVAL

Norman Adams, Daniela Marquez, Gregory Wakefield
Electrical Engineering and Computer Science
University of Michigan
1101 Beal Ave., Ann Arbor, 48109, USA
nhadams@umich.edu

ABSTRACT

For melodic theme retrieval there is a fundamental tradeoff between retrieval performance and retrieval speed. Melodic representations of large dimension yield the best retrieval performance, but at high computational cost, and vice versa. In the present work we explore the use of iterative deepening to achieve robust retrieval performance, but without the accompanying computational burden. In particular, we propose the use of a smooth pitch contour that facilitates query and target representations of variable length. We implement an iterative query-by-humming system that yields a dramatic increase in speed, without degrading performance compared to contemporary retrieval systems. Furthermore, we expand the conventional iterative framework to retain the alignment paths found in each iteration. These alignment paths are used to adapt the alignment window of subsequent iterations, further expediting retrieval without degrading performance.

Keywords: Iterative deepening, DTW, melody retrieval.

1 INTRODUCTION

Melodies, similar to other time-series representations, present a particular challenge for database retrieval; the temporal elasticity of musical performance prevents the use of a direct similarity measure, such as Euclidean distance [1]. This problem is acute for casually sung melodies, which are often the focus of MIR applications [2, 3, 4]. To address the problem, a wide variety of dynamic alignment methods have been developed that account for a time warping between tokens, thus providing a more robust measure of similarity in the face of human performance factors [5, 6, 7, 8, 9]. Unfortunately, all such alignment methods present a computational burden in comparison to direct similarity measures. Dynamic alignment methods have computational complexity $O(N^2)$, where N is the length of the tokens, rather than $O(N)$ for direct measures. Accordingly, it is desirable to keep the length, or dimension, of tokens as small as possible. However, this comes at the expense of degraded retrieval performance.

In the present work we propose the use of iterative deepening (ID) coupled with dynamic time warping (DTW) to expedite melody retrieval without degrading performance. Iterative deepening uses a scalable representation to retrieve target tokens with successively more accurate similarity measures [10]. Only a fraction of the targets are retained after each iteration, hence the probability of false dismissals increases. However, by allowing small increases in false dismissals, iterative deepening can greatly speed retrieval. We describe and evaluate iterative deepening in the context of a query-by-humming (QBH) system, and present results showing a dramatic speed increase with a negligible increase in false dismissals. Furthermore, we propose a novel method to further expedite performance by retaining alignment information for future iterations and using an adaptive alignment window. In so doing, we achieve an even greater speed increase over conventional methods with only minimal detriment to retrieval performance.

The remainder of this section provides background information about melodic representations for QBH and the tradeoff between retrieval performance and speed. Section 2 presents in turn dynamic alignment, iterative deepening, and adaptive iterative deepening. Section 3 describes our methods for evaluation, along with results and discussion. Section 4 concludes the paper.

1.1 Background

The MIR community has explored many melodic representations for music retrieval; sequences of notes [2, 7, 11] and various quantizations thereof [9, 12, 13, 14], smooth pitch contours[1] [5, 6, 15], sequences of histograms [5, 8], and hidden Markov models (HMM) [4, 16]. Such research has focused primarily on the retrieval performance a given representation affords, without considering the retrieval speed in great detail. For modestly sized target databases,

[1]In the present work, the "pitch contour" is defined as the output of a pitch tracking algorithm. See Fig. 1 for a sample pitch contour. Other authors use "pitch contour" to refer to a coarsely quantized sequence of pitch differences of a note sequence [13].

as are often used for QBH experiments, retrieval speed is not a problem. But as the target database becomes massive expedient retrieval becomes a more dif cult and desirable goal. While note-based melodic representations have tended to dominate research in query-by-humming systems [2, 9, 11, 12], there is growing interest in smooth pitch contour representations [5, 6, 15, 17, 18]. This is due largely to the dif culty of transcribing casually sung melodies, where detecting note onsets is especially error-prone [4, 11, 19]. Unfortunately, while contour representations yield promising results, they present a large computational burden [5, 6, 17, 15].

Previous work by the authors comparing different melodic representations found that representations that yield the most robust retrieval performance also yield the slowest retrieval time [5]. In particular, we found a smooth pitch contour yields the best performance on large databases, but only for relatively long contours, with length (i.e. dimension) $N \sim 100$. In contrast, a simple note representation has dimension $N \sim 10$, which facilitates relatively rapid retrieval but gives mediocre performance on large databases. Furthermore, we found that using a pitch contour with small dimension, $N \sim 10$, yields performance similar to conventional note representations.

The tradeoff between robust and rapid retrieval is common in database problems. As such, the database community has begun exploring methods to speed dynamic alignment and retrieval of long time series. Techniques include lower bounds [20], alignment and continuity constraints [6], indexing [16, 21], and iterative methods [10, 22]. In the present work, we adapt the latter to the unique challenges of MIR by using a melodic representation that is readily decimated to arbitrary length.

2 METHODS

2.1 Representation

We restrict our attention to smooth pitch contour representations of melody. For a sung or hummed query, the pitch contour is estimated using a time-domain method [11, 23]. This algorithm computes the autocorrelation for overlapping frames of recorded data with constant step-size 10ms. A set of candidate peaks is selected for every frame and the Viterbi algorithm is used to construct a smooth contour. The values of fundamental frequency are then converted to MIDI pitch[2]. The estimated pitch contour contains gaps where no pitch is estimated. These gaps are ‘ lled in’ by extending the end of each pitched region to the start of the next. The autocorrelation values are used in extending the regions so as to prevent incongruous pitch uctuations [5]. Direct use of the estimated contour yields a representation with prohibitively large dimension, $N > 1000$. Hence the contours are decimated to yield a dimension $10 < N < 200$ [1, 15]. In the present work, queries are individually resampled such that the dimension N is constant for all queries.

Fig. 1 shows a sample pitch contour for the main theme from My Bonnie lies over the ocean with $N =$ 80. Included in Fig. 1 is the piecewise constant ideal contour for the same theme stored in the target database. Note that the target contour has been time-scaled to have equal duration as the query. The next section details the similarity measure used for comparing query and target contours.

[2]The real-valued MIDI pitch number p is related to a signal's fundamental frequency in Hz, f_0, as $p = 12\log_2(f_0/261) + 60$.

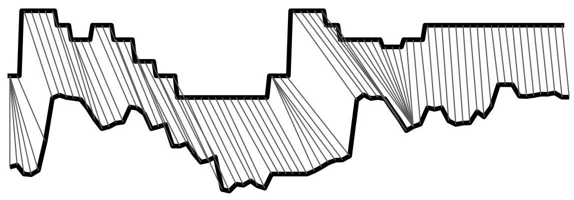
Figure 1: Sample alignment for the melody sung to the lyrics My Bonnie lies over the ocean. My Bonnie lies over the sea. . The top contour is the target, and the bottom contour is the query.

2.2 Time Series Alignment

Query-by-humming systems typically operate by computing a measure of similarity between a sung query and every theme in a database of targets. The target themes are then rank ordered by similarity. In the present work we use DTW to align the query to each target in the database, interpreting the alignment cost as a measure of distance between the query and target [5]. For a given query contour with length N, every target theme is rst time-scaled to yield a piece-wise constant contour of length $N' \approx N$. The mean pitch difference between the query and target is then subtracted, yielding query contour $\mathbf{Q} = (q_1, q_2, \cdots q_N)$ and target contour $\mathbf{T} = (t_1, t_2, \cdots t_{N'})$. To account for a time-warping of the query relative to the target, the elements of the query contour must be aligned to the appropriate elements of the target. A sample alignment is shown in Fig. 1 for a query/target pair of the main theme from the traditional tune My Bonnie lies over the ocean. .

For a given continuity and cost scheme, a brute-force search of all possible alignments is computationally prohibitive, hence dynamic programming is employed so as to compute the cost of only the optimal alignment [24]. Let $\mathbf{\Gamma} = [\gamma_{n,k}]$ be an $N \times N'$ matrix of minimum pre x alignment costs; $\gamma_{n,k}$ is the minimum alignment cost for $(q_1 \cdots q_n)$ and $(t_1 \cdots t_k)$. Let a warping path be given by $\mathbf{w} = (w_1, w_2, \cdots w_T)$, where $w_t = (n, k)$ indicates that q_n is aligned with t_k. The warping path must adhere to several constraints to be physically meaningful. The path must begin with the rst elements of the query and target contour, and end with the last elements; $w_1 = (1, 1)$ and $w_T = (N, N')$. The path must be monotonic nondecreasing and adhere to a local continuity constraint. Starting in the lower left corner of $\mathbf{\Gamma}$, every element of $\mathbf{\Gamma}$ is found recursively by

$$\gamma_{n,k} = \min \begin{pmatrix} \gamma_{n-1,k} + \zeta \\ \gamma_{n,k-1} + \zeta \\ \gamma_{n-1,k-1} \end{pmatrix} + |q_n - t_k| \qquad (1)$$

where $\zeta \geq 0$ is an additive cost penalty applied to favor more direct alignments. To speed computation and prevent extreme warpings, the alignment path is restricted to be near the main diagonal $n = k$. In particular, $\forall n, k : |n - k| > N/5 \rightarrow \gamma_{n,k} = \infty$. The nal alignment cost for $\mathbf{Q}$ and $\mathbf{T}$ is given by $\gamma_{N,N'}$. The computational complexity of this procedure is $O(N^2)$. Note that the nal alignment cost is not normalized by T, which would render the alignment suboptimal for the cost scheme in (1). However, the effect of suboptimal alignment on *retrieval* performance remains an unanswered question. Indeed, we have informally explored alignment procedures, not presented here, that yield suboptimal alignments but slightly improved retrieval performance.

2.3 Iterative Deepening

Consider the following simple QBH experiment, which highlights the tradeoff between retrieval performance and N. For any given N, we are interested in searching over a fraction of the target database. Let us de ne a 'success threshold' as the top percentage of the ordered target database that the correct theme must be in for the retrieval to be considered a success[3]. We then compute the fraction of queries for which the retrieval is successful as a function of the 'success threshold', and label this fraction the 'success rate' [14]. Fig. 2 gives the retrieval performance for a DTW QBH system for $N = \{12, 14, 20, 30, 50, 100, 150\}$. The query and target databases used to generate this gure are the same as those detailed in Section 3. The abscissa represents the success threshold, and the ordinate gives the fraction of queries for which the retrieval is successful. For example, for a 10% success threshold, the $N = 20$ representation yields a success rate of 0.86.

For this experiment the target database contained 1000 themes. Hence the left edge of each curve, which gives the success rate for a 0.1% threshold, is equivalent to the classi cation accuracy of the system. The point corresponding to a 1% threshold gives the fraction of queries for which the correct theme was in the top ten themes returned by the retrieval system. It is evident from the gure that the best classi cation accuracy, 85%, is achieved only for $N = 150$. Extending the 85% 'success' rate across the gure highlights the '85% performance capacity' versus dimension. That is, $N = 14$ may yield mediocre classi cation accuracy, but if all that is desired is that the correct theme is in the top 20% of the returned targets, then $N = 14$ would appear to perform reasonably well. While this may not be a good stopping point for our retrieval system, we have made the problem somewhat simpler by determining that 800 of the 1000 targets can be discarded. This motivates a retrieval system that makes multiple passes through the target database and removes, after each iteration, as many targets as possible subject to a bound on false dismissals. Numerous iterative techniques have been proposed, which often make strictly two passes through the database [14, 16, 20]. In the present work

[3]For example, consider a target database with 100 themes. If the rank of the correct theme for a given query is 7, then the retrieval is considered a success if the success threshold is 10%, but a failure if the success threshold is 5%.

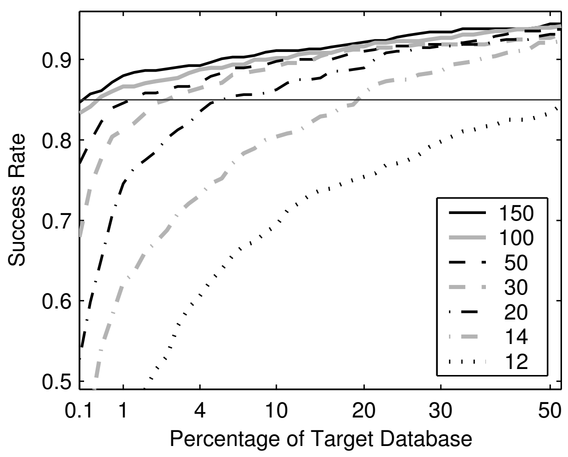

Figure 2: Retrieval performance for varying dimension, N.

we explore a retrieval architecture amenable to a variable number of iterations [10].

Consider a direct DTW system, which must align a query of length N_M with the full set of targets. Let the computational cost for the direct method be $c_F = C \cdot N_M^2$ for some constant C. Rather than performing the rather costly length N_M alignment with all targets, the iterative method uses a sequence of increasing representation lengths, $\mathbf{N} = (N_1, N_2, \cdots N_M)$. The query is rst aligned with all the targets using length N_1. After each iteration, the alignment costs are sorted and only the targets yielding the smallest alignment costs are retained for subsequent iterations. Let r_i be the fraction of the target database retained for the i^{th} iteration, $\mathbf{r} = (1, r_2, \cdots r_M)$. Clearly, the speci c choice of $(r_2, \cdots r_M)$ has considerable impact on the ultimate retrieval performance and speed. The computational cost for the iterative method is[4]

$$c_{ID} = C \cdot (N_1^2 + r_2 N_2^2 + \cdots + r_M N_M^2). \qquad (2)$$

Hence the computational cost for the iterative method relative to the direct method is

$$\rho = \frac{c_{ID}}{c_F} = \left(\frac{N_1}{N_M}\right)^2 + r_2 \left(\frac{N_2}{N_M}\right)^2 + \cdots + r_M. \qquad (3)$$

2.4 Adaptive Alignment Constraints

While it is clear that iterative deepening can greatly expedite many database retrieval tasks, the method is wasteful in the sense that the alignments themselves are discarded after each iteration. Computing the optimal alignment for the same query/target pair for multiple representation lengths is somewhat redundant. As N increases, the overall shape of the optimal alignment is unlikely to change, but rather the ne detail of the optimal alignment will be resolved. This observation implies that the iterative deepening framework can yield even faster performance if information about prior alignments is incorporated into each iteration. In Section 2.2 we found that an

[4]We are neglecting the cost due to the additional sort operations the iterative method requires, which is small compared to the alignment cost.

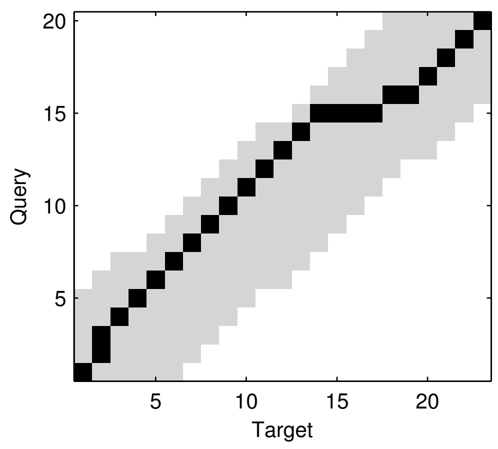

Figure 3: Sample alignment for $N = 20$.

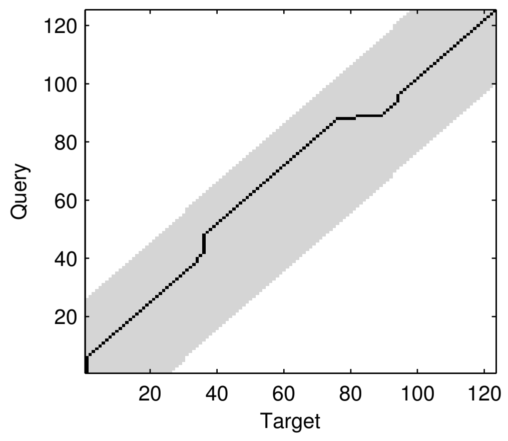

Figure 4: Sample alignment for $N = 120$.

alignment window of 20% yields good retrieval performance. By de ning the alignment window relative to the previous alignment rather than the main diagonal, we can reduce the width of the window considerably without sacri cing alignment performance, thus quickening retrieval. A similar idea was recently proposed in [22].

Adaptive alignment windows are perhaps best described with an example. Consider the alignment of $N = 20$ contours for the same query and target shown in Fig. 1. Fig. 3 shows the 20% alignment window in grey, and the nal alignment path in black. We then realign the query and target for $N = 120$. We do this in two ways, with and without prior alignment. Without using the prior alignment we cannot safely reduce the alignment window width below 20%. This case is shown in Fig. 4. Note that the basic shape of the alignment in Figs. 3 and 4 is the same. Instead of using a 20% window centered around the main diagonal, we can use a narrow 10% window centered around the shape of the $N = 20$ alignment. This case is shown in Fig. 5. The alignment window in Fig. 5 is computed by scaling the $N = 20$ alignment to the $N = 120$ matrix and vertically stretching the the window. Note that the nal alignments shown in Figs. 4 and 5 are identical. This implies that we can use the prior alignment

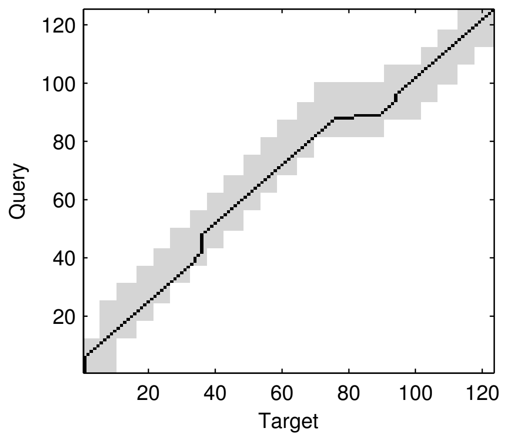

Figure 5: Sample alignment for $N = 120$ using a constrained alignment window from $N = 20$.

to halve the size of the current alignment window, without degrading the quality of the nal alignment. By de nition, the adaptive alignment constraint cannot expedite the rst iteration of retrieval, but the speed of all subsequent iterations will be doubled. Hence the computational cost of the adaptive iterative deepening (AID) relative to the direct method is

$$\rho' = \frac{c_{AID}}{c_F} = \left(\frac{N_1}{N_M}\right)^2 + \frac{1}{2}\left(r_2\left(\frac{N_2}{N_M}\right)^2 + \cdots + r_M\right). \tag{4}$$

3 EVALUATION

We evaluate the performance of the iterative deepening in the context of a query-by-humming system. Previous work by the authors found that retrieval performance plateaus for $N > 150$, and falls to chance for $N < 10$. Hence an iterative retrieval system that begins with $N \sim 10$ and ends with $N \sim 150$ would be a judicious choice. We implement an ID-QBH system using $M = 3$ iterations, with $N_1 = 14$, $N_2 = 32$ and $N_3 = 144$. The values of $mathbfN$ were determined experimentally. N_1 is the most critical of the three, for smaller values yield substantially worse performance and larger values yield substantially slower retrieval. The precise values of N_2 and N_3 are not critical, although we found that N_2 should generally be closer to N_1 than N_3. We found using $M > 3$ did not improve performance, whereas performance for $M = 3$ does improve performance somewhat over $M = 2$. For comparison we present results for conventional DTW QBH systems using $N = 14$ and $N = 144$. The next two subsections describe the query and target databases used for evaluation. Section 3.3 details the choice of r_2 and r_3, the fraction of the target database retained for the second and third iterations, respectively. Results are then presented in section 3.5.

3.1 Query Database

For performance evaluation, we employ a query test set containing many sample queries of fourteen popular tunes, from the Beatles' Hey, Jude to Richard Rodgers' Sound of Music. A total of 480 queries were collected from fteen participants in our study. Each participant was asked to sing a familiar portion of a subset of the fourteen tunes four times each. The participants had a variety of musical backgrounds; some had considerable musical or vocal training while most had none at all. Participants were instructed to sing each query as naturally as possible using the lyrics of the tune[5]. The queries are monophonic, 16 bit recordings sampled at 44.1 kHz and resampled to 8 kHz to reduce processing time. The queries are between 4 and 20 seconds in length. All data were collected in a quiet classroom setting and participants were free to progress at their own pace.

Note that for all fourteen melodies, every participant sang the same set of measures. For a real-world QBH system, this is an unreasonable assumption. This problem can be ameliorated by allowing for zero-cost insertion and deletion steps at the beginning and end of the alignment[6], similar to [4]. Another common practice is to include multiple themes for each tune in the target database [17], increasing the size of the target database.

3.2 Target Database

Measuring the retrieval performance of our QBH methods on a target database of only the fourteen themes for which we have sample queries would not indicate how well the methods would perform in a broader context. Accordingly, we augment the target database with extra themes not included in the query test set. We ensure that the additional targets are suf ciently similar to the 14 'authentic' targets by building a Markov model that generates 'synthetic' targets with similar rst-order statistics as the authentic themes [5, 25]. In particular, we generate 986 synthetic themes, yielding a target database of 1000 themes. We can verify that the synthetic themes are similar to the authentic themes by considering the ratio of authentic to synthetic targets that are retained after each iteration of system retrieval. We nd that for the i^{th} iteration, the number of incorrect authentic themes included in the iteration is about $r_i \cdot \frac{13}{999}$. Hence each authentic theme is as similar, in the sense of DTW alignment cost, to the other 13 authentic themes as it is to the 986 synthetic themes. While many of the synthetic themes are musically unsatisfying, they clearly demonstrate enough similarity to the authentic themes so as to challenge the retrieval system. We note from the outset that using synthetic targets limits how the results can be interpreted. However, we are concerned with relative trends rather than absolute performance.

3.3 Performance-Speed Tradeoff

Having xed the number of iterations, $M = 3$, and dimensions, $\mathbf{N} = (14, 32, 144)$, we must choose r_2 and r_3 to completely specify the system. Clearly, as r_2 and r_3 increase, both the retrieval performance and the retrieval time increase. We seek to minimize r_2 and r_3 without degrading retrieval performance. This can be done by estimating the distribution of approximation error at each representation length, and using this to estimate the probability of false dismissal for each r_i [10]. In the present work we simply examine the net retrieval performance versus r_2 and r_3, and choose the smallest values that do not substantially degrade retrieval. Figures 6 and 7 highlight the tradeoff between retrieval performance and speed.

Fig. 6 gives the mean reciprocal rank (MRR) versus r_2 and r_3, and Fig. 7 gives ρ versus r_2 and r_3. In both plots the minimum value of r_2 and r_3 is 0.1%, and the maximum value of r_2 is 50% and the maximum value of r_3 is 10%. Note that for $r_2 = 0.1\%$, the MRR equals the classi cation accuracy of a direct $N = 14$ system, and for $r_3 = 0.1\%$ the MRR equals the classi cation accuracy of a direct $N = 32$ system. From (3) we know that as r_2 and r_3 approach zero, ρ converges to 0.0095, which is readily apparent in Fig. 7. While MRR decreases with r_2 and r_3, a distinct threshold is evident; so long as r_2 is greater than $10-20\%$ and r_3 is greater than $1-2\%$ the MRR is approximately constant at 0.85. Accordingly, we set $r_2 = 20\%$ and $r_3 = 2\%$. By (3) and (4), this yields $\rho = 0.039$ and $\rho' = 0.024$. Hence the iterative system will be about 25 times faster than conventional DTW with $N = 144$, and the adaptive iterative method will be about 41 times faster.

3.4 Results

Performance results are summarized in Tables 1 and 2. From the two tables it is clear that the iterative QBH systems yield much faster retrieval than a direct $N = 144$ DTW QBH system, without degrading performance substantially. The iterative systems successfully circumvent the tradeoff between retrieval performance and speed that 'one-pass' retrieval systems are subject to, yielding a system with the performance of the $N = 144$ system and the speed of the $N = 14$ system.

Table 1 reports three retrieval performance metrics for four QBH systems. The top two systems listed are direct implementations of the DTW system described in section 2.2 for $N = 14$ and $N = 144$. The third system listed in the iterative method described in section 2.3, and the fourth system listed is the adaptive iterative method described in section 2.4. The rst performance metric listed is classi cation accuracy (CA), the fraction of queries for which the correct theme is rank one in the retrieved list of themes. The second performance metric listed is the top 10 fraction (Top 10), the fraction of queries for which the correct theme is amongst the top ten themes returned. The third performance metric listed is the mean reciprocal rank (MRR), the average inverse rank of the correct theme.

From Table 1, it is clear DTW for $N = 144$ outperforms the case when $N = 14$ in all three performance measures. In terms of CA and MRR, the higher dimension yields considerably better performance. In terms of the top 10 fraction however, the case of $N = 14$ does not perform as poorly, yielding a top 10 fraction of 0.80 com-

[5]This contrasts substantially from the common practice of having participants sing isolated pitches on a neutral vowel, requiring participants to perform note segmentation [11].

[6]Although, for this approach to be effective, the alignment window must be made considerably larger near the beginning and end of the alignment

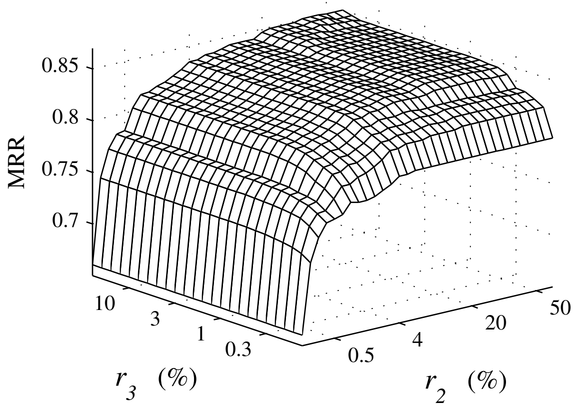

Figure 6: Mean reciprocal rank.

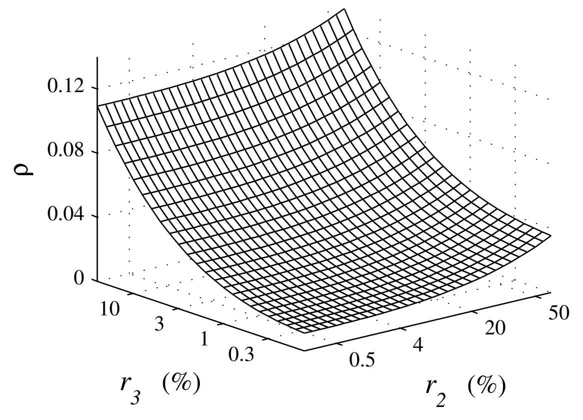

Figure 7: Computation gain.

Table 1: Retrieval performance of four QBH systems

System	CA	Top 10	MRR
DTW, $N = 14$	0.663	0.803	0.723
DTW, $N = 144$	0.840	0.869	0.851
Iterative DTW	0.837	0.862	0.847
Adt. It. DTW	0.835	0.860	0.845

Table 2: Computation cost of four QBH systems

System	Cells (1000's)	Time (s)
DTW, $N = 14$	93	1.61
DTW, $N = 144$	7720	106
Iterative DTW	329	4.87
Adt. It. DTW	208	3.49

pared to 0.86 for the case of $N = 144$. While a small representation may not yield good CA or MRR, apparently it can determine which set of 10 targets are likely to contain the correct theme. Comparing the retrieval performance of the two iterative systems to that of the direct systems, we see that the two iterative systems yield retrieval performance virtually identical to that of the direct system. All three metrics fall modestly from the direct $N = 144$ system to the iterative system, and further still for the adaptive iterative system. The performance drop is negligible compared to the performance difference between the direct $N = 14$ and $N = 144$ systems, however.

While the two iterative systems yield retrieval performance virtually identical to the direct $N = 144$ system, the retrieval speed of the iterative systems is closer to that of the direct $N = 14$ system. Table 2 gives both the average total number of alignment cells, $\gamma_{n,k}$, that must be computed for each query, and the average total retrieval time per query[7]. The direct $N = 144$ yields an average per query retrieval time of 106 seconds. This system is far too slow to deploy directly on a massive target database. The other three systems all yield a retrieval time of less than 5 seconds. The direct $N = 14$ system yields the fastest retrieval, but at the expense of good retrieval performance. The iterative system yields an average retrieval time of 4.9 seconds, 21 times faster than the direct $N = 144$ system. The adaptive iterative system yields an average retrieval time of 3.5 seconds, 30 times faster than the direct $N = 144$ system.

[7]The retrieval time listed is the actual retrieval time we recorded for our MATLAB implementation running on a laptop PC, with a 1.8 GHz CPU

3.5 Discussion

For our choice of M, $\mathbf{N}$, and $\mathbf{r}$, (3) predicts that the iterative method is 25 times faster than the direct implementation, and (4) predicts that the adaptive iterative method is 41 times faster. The disparity between the predicted and observed computational cost for the iterative method is due to the extra sort operations the iterative method requires. The disparity between the predicted and observed computational cost for the adaptive iterative method is larger. In this case the disparity is also due to the large memory requirements of the adaptive iterative method. After each iteration, the optimal alignment paths must be stored for use in subsequent iterations. The alignment paths consume a considerable amount of memory, and ultimately delay retrieval. When measured in terms of the average total number of cell visits per query, the speed increases for the iterative method and adaptive iterative method are 23 and 37, respectively.

From Fig. 6 it is evident that for this target database, if all that is desired is that the correct theme be in the top 10 of returned themes, then the nal $N = 144$ iteration is largely unnecessary. That is, the MRR plateaus for $r_3 > 1\%$. Hence retrieval performance can be further expedited in this case by simply returning the results of the $N = 32$ iteration. Indeed, for the iterative methods the nal $N = 144$ iteration simply determines the order of the 20 themes returned by the $N = 32$ iteration. However, we have found that as the target database grows, the nal $N = 144$ iteration becomes critical [5]. As the target space becomes more densely populated with themes, increasing the dimension of the target space plays a crucial role in retrieval. Hence iterative methods can greatly facilitate QBH systems with massive target databases, where having a large melodic representation is critical for robust retrieval, but too slow for direct implementation. The iterative methods allow for the retrieval time to grow less than linearly as the target database size increases. Furthermore, constraining the alignment window size as much as possible is critical for deploying QBH systems on massive target databases, for as the alignment window shrinks the computational complexity of the alignment converges to $O(N)$[8] [20].

A 30 fold speed increase over direct methods, while substantial, may not be suf cient for scaling DTW methods to truly massive databases, for the computational complexity is still linear with the target database size. However, we have conducted informal experiments that indicate that $\mathbf{r}$ can be reasonably reduced as the target database grows. Hence the total computational complexity of this method may effectively be less than linear with target database size.

Finally, we note that retrieval performance for all systems can be increased considerably by removing the sample queries of three subjects from our test database. For example, the MRR of the direct $N = 144$ system increases to 0.93 if these queries are removed. These queries are very inaccurate and some are virtually monotone; the lyrics are their only recognizable feature. It is unclear that any QBH system should be designed to accommodate such queries without explicitly modeling singer production and transcription error [4, 7].

4 CONCLUSION

Previous work by the authors explored the tradeoff between retrieval performance and retrieval speed. In the present work we proposed iterative deepening as a technique to circumvent this tradeoff, in particular for large target databases. We employed a smooth pitch contour that is amenable to variable representation length. An iterative query-by-humming system was implemented that makes three passes through the target database, removing as many targets as possible with each iteration without introducing undue false dismissals. We also expanded the conventional iterative framework to retain the alignments found in each iteration. These alignments are used to adapt the alignment window of subsequent iterations, further expediting retrieval. We nd that iterative deepening can speed retrieval by a factor of 25, and the adaptive iterative method can speed retrieval by a factor of 40, both without signi cantly degrading retrieval performance.

ACKNOWLEDGEMENTS

This research was supported by a CARAT/Rackham Interdisciplinary Fellowship and a grant from the ITR project of the NSF. We would also like to thank Joe Heremans for his assistance with query and target collection.

[8]Other authors define the window size as a separate constant rather than as a fraction of N [22]. Although this has no effect on the exact computational cost, it does yield a complexity that is, strictly speaking, linear in N.

REFERENCES

[1] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems 3(3)*, 2000.

[2] R. J. McNab, L. A. Smith, and I. H. Witten et al. Towards the Digital Music Library: Tune Retrieval from Acoustic Input. *Proc. ACM Digital Libraries Conference*, 1996. Bethesda, MD.

[3] J. S. Downie and M. Nelson. Evaluation of a Simple and Effective Music Information Retrieval Method. In *Proc. ACM SIGIR*, pages 73 80, 2000.

[4] C. Meek and W. Birmingham. Johnny can't sing: A comprehensive error model for sung music queries. *Proc. ISMIR*, October 2002.

[5] N. H. Adams, M. A. Bartsch, J. Shiffrin, and G. H. Wake eld. Time Series Alignment for Music Information Retrieval. *Proc. ISMIR*, 2004. Barcelona, Spain.

[6] N. Hu and R. Dannenberg. A Comparison of Melodic Retrieval Techniques Using Sung Queries. *Joint Conf. on Digital Libraries*, 2002.

[7] A. Pikrakis, S. Theodoridis, and D. Kamarotos. Recognition of Isolated Musical Patterns Using Context Dependent Dynamic Time Warping. *IEEE Trans. Speech and Audio Processing*, 11(3):175 183, May 2003.

[8] J. Song, S.Y. Bae, and K. Yoon. Mid-Level Music Melody Representation of Polyphonic Audio for Query-by-Humming System. *Proc. ISMIR*, 2002. Paris, France.

[9] M. Grachten, J.-L. Arcos, and R. Lopez de Mantaras. Melodic Similarity: Looking for a Good Abstraction Level. *Proc. ISMIR*, 2004. Barcelona, Spain.

[10] S. Chu, E. Keogh, D. Hart, and M. Pazzani. Iterative Deepening Dynamic Time Warping for Time Series. *Second SIAM International Conference on Data Mining*, 2002.

[11] N.H. Adams, M.A. Bartsch, and G.H. Wake eld. Note Segmentation and Quantization for Music Information Retreival. to appear *IEEE Trans. Speech and Audio Processing*, January 2006.

[12] A. Ghias, J. Logan, D. Chamberlin, and B.C. Smith. Query by humming: Musical information retrieval in an audio database. In *ACM Multimedia*, pages 231 236, 1995.

[13] Y. E. Kim, W. Chai, R. Garcia, and B. Vercoe. Analysis of a Contour-based Representation for Melody. In *Proc. ISMIR*, October 2000.

[14] R.B. Dannenberg and N. Hu. Understanding Search Performance in Query-by-Humming Systems. *Proc. ISMIR*, 2004. Barcelona, Spain.

[15] D. Mazzoni and R. B. Dannenberg. Melody Matching Directly from Audio. *Proc. ISMIR*, 2001. Bloomington, IN.

[16] H. Jin and H.V. Jagadish. Indexing Hidden Markov Models for Music Retrieval. *Proc. ISMIR*, 2002. Paris, France.

[17] R. B. Dannenberg, W. P. Birmingham, and G. Tzanetakis et. al. The MUSART Testbed for Query-By-Humming Evaluation. In *Proc. ISMIR*, 2003.

[18] Y. Zhu and D. Shasha. Warping Indexes with Envelope Transforms for Query by Humming. In *Proc. International Conference of Management of Data (SIGMOD)*, San Diego, CA, 2003.

[19] R. J. McNab and L. A. Smith. Evaluation of a Melody Transcription System. *IEEE Int. Conf. on Multimedia and Expo, 2000*, 2:819 822, 2000.

[20] C.A. Ratanamahatana and E. Keogh. Everything you know about Dynamic Time Warping is Wrong. In *Proc. 3rd Workshop of Mining Temporal and Sequential Data*, 2004.

[21] S.W. Kim, S. Park, and W.W. Chu. Ef cient processing of similarity search under time warping in sequence databases: an index-based approach. *Information Systems*, 29(5):405 420, 2004.

[22] S. Salvador and P. Chan. FastDTW: Toward Accurate Dynamic Time Warping in Linear Time and Space. In *Proc. KDD Workshop on Mining Temporal and Sequential Data*, 2004.

[23] P. Boersma. Accurate Short-Term Analysis of the Fundamental Frequency and the Harmonics-to-Noise Ratio of a Sampled Sound. In *Proc. Institute of Phonetic Sciences of the University of Amsterdam*, volume 17, pages 97 110. 1993.

[24] L.R. Rabiner and B.H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, Upper Saddle River, NJ, 1993.

[25] A. Ito, S.-P. Heo, M. Suzuki, and S. Makino. Comparison of Features for DP-Matching Based Query-by-Humming System. *Proc. ISMIR*, 2004. Barcelona, Spain.