Time Series Representations for Music Information Retrieval

Norman H. Adams

May 5, 2004

Abstract

Time series representations are common in MIR applications such as query-by-humming, where a sung query might be represented by a series of 'notes' for database retrieval. While such a transcription into (pitch, duration) pairs is convenient and musically intuitive, there is no evidence that it is an optimal representation. The present work explores three time series representations for sung queries: a sequence of notes, a 'smooth' pitch contour, and a novel sequence of pitch histograms. Dynamic alignment procedures are described for the three representations. Multiple continuity constraints are explored and a modified dynamic alignment procedure is described for the histogram representation. We measure the performance of the three representations using a collection of naturally sung queries applied to a target database of varying size. The results show that the note representation lends itself to rapid retrieval whereas the contour representation lends itself to robust performance. The histogram representation yields performance nearly as robust as the contour representation, but with computational complexity similar to the note representation.

1 Introduction

The ever increasing number of digital audio files available on the internet motivates the idea of web searches that use acoustic signals as query input rather text strings. Music information retrieval (MIR) is a burgeoning area of research within the broader field of "content-based" information retrieval. In particular, the MIR community is exploring techniques to allow a user to search a database of music using only a brief snippet of recorded audio; no information about a song's composer, performer, name or lyrics is required. Of particular interest are systems that allow the user to simply sing, or hum, a short melodic phrase. Systems that support this sort of interaction are known as query-by-humming (QBH) systems. As for any information retrieval task, the sung query must be coded to allow for efficient retrieval. The choice of query representation is of paramount importance; compact representations lend themselves to rapid retrieval whereas detailed representations lend themselves to accurate retrieval.

Time series representations are ubiquitous in information retrieval applications [18,43]. MIR is no exception; sequences of notes, pitch estimates, or MFCCs, for example, are common [7,9,25]. It is well established that for comparing two time series, a direct comparison, such as the Euclidean distance, yields a brittle metric [17,43]. A similarity metric that is robust to elastic shifts and scales

of the time index is required. This has reinvigorated interest in dynamic time warping (DTW), string-matching, and other efficient time series alignment methods within the IR community. All such alignment procedures have complexity $O(NK)$, where N and K are the lengthes of the two sequences. Accordingly, it is desirable to keep the length, or dimension, of the representation as small as possible. However, this often comes at the expense of retrieval performance.

In the present work we compare the relative merits of three time series representations for sung queries. Two representations have been previously proposed: a pitch contour[1] [25], and a sequences of notes [9, 28, 32]. A unified presentation of the alignment procedure is given for the two representations. For the contour representation we explore multiple continuity constraints and find that judicious slope constraints improve performance considerably. We also consider two methods for reducing the dimensionality of the contour representation: decimation and piecewise linear approximation. For the note representation, multiple note estimators are considered. We include multiple implementations of the contour and note representations so as to demonstrate the range of performance possible using either representation; we are interested in the relative merits of each representation rather than a specific implementation.

Finally, we propose a novel sequence of pitch histograms as an alternative query representation. Unlike other pitch histogram representations [13, 39], which were proposed to allow for errors in pitch detection, we employ pitch histograms to eliminate ambiguous timing information, thus quickening the alignment procedure. This representation violates the local comparison constraint of the common DTW algorithm, hence a modified dynamic alignment procedure is presented.

We evaluate the performance of the three representations in the context of a query-by-humming system. Early QBH successes by [11, 15, 28] have inspired numerous alternative techniques [9, 13, 32]. While many exciting and novel ideas have been explored, results are often inconclusive and difficult to generalize [9, 10]. In the present work we focus on relative performance trends between the three representations rather than the absolute performance of any one. We apply our methods to a variable database of themes, similar to [9]. In so doing, we observe some interesting trends between the different methods.

As for any experiment, numerous simplifications are made to facilitate measurement and eliminate distracting details. Our hope is that by considering how the performance decreases as target database size increases we arrive at a measure of how the methods would perform in a broader context. That is, many of the complicating factors that detriment QBH performance in a real-world scenario are, in effect, equivalent to simply expanding the size of the target database.

1.1 Background & Motivation

To date, most QBH systems first transcribe the sung melody into a sequence of 'notes'[2], and use this representation to search a database of similarly represented target themes [2, 5, 11, 22, 27, 28]. While a note representation may be musically intuitive, there is no hard evidence to imply that it is optimal in any specific technical sense. Neglecting the musical justification for notes, the transcription stage of these QBH systems can be viewed simply as reducing the dimension of the query to a manageable size for database searching. For example, the dimension of a 10 second

[1] In the present work, the "pitch contour" is defined as the output of a pitch track algorithm. See Fig. 4 for a sample pitch contour. Other authors use "pitch contour" to refer to a coarsely quantized sequence of pitch differences of a note sequence, as in Parson's directory of themes, i.e., "UDUDRU" for example [31].

[2] Strictly speaking, many systems do not estimate the note sequence, but rather a coarsely quantized sequence of pitch differences. Furthermore, many systems do not code note duration [29].

query recorded sampled at 8 kHz is 80,000. Clearly, performing a $O(NK)$ alignment for $N \approx K \approx 80,000$ (even if an appropriate cost scheme could be found) is computationally prohibitive. Transposed queries on the other hand, typically contain between 8 and 20 notes, a more reasonable size. For a given query, this $O(NK)$ alignment procedure, often referred to *string-matching*, must be performed for every theme in the target database. The themes in the target database are rank ordered using the alignment cost for each target as a measure of dissimilarity. It is essential that the computational complexity of comparing a query to a target be kept to a minimum. Other methods have been proposed for comparing query and target note representations, such as hidden Markov models (HMM) [29, 38]. While HMMs' have been found to yield good results in some situations, they are not considered in this work. Note representations and string-matching are discussed in detail in Section 4.

Some researchers have begun to reconsider the reliance on sung melody transcription as the de facto method for reducing query dimensionality. In particular, Mazzoni and Dannenberg have proposed direct use of the pitch contour in QBH systems [24, 25]. In their work Mazzoni use *dynamic time warping* (DTW) to align the pitch contour of the sung query with the pitch contour for every melody in the target database. String-matching and DTW are largely equivalent, both employ *dynamic programming* (DP) to efficiently search for an optimal alignment between to time series based on fixed continuity constraints and cost schemes. String-matching is typically applied to sequences of discrete symbols, whereas DTW is applied to real-valued sequences. Furthermore, string-matching cost schemes often explicitly allow for 'inserting' or 'deleting' an element from one sequence, in which case this element is not matched to any element in the other sequence. This is not the case for DTW cost schemes, every element from the query sequence must be matched to an element from the target sequence. For DTW a match cost is associated with every alignment step, for string-matching 'non-diagonal' steps have separate insert/delete costs and no match cost. Contour representations and DTW are discussed in Section 3.

Another interesting alternative was presented in [39]. In their work Song et al explore the use of a sequence of pitch histograms as a mid-level representation for polyphonic music. They use pitch histograms rather than scalar pitch estimates to circumvent the difficulty of extracting the perceived melody from a cluttered acoustic signal. While the proposed representation they used has many potential benefits many details of their implementation were ad hoc and the results were somewhat inconclusive. In the present work we also consider using a sequence of pitch-histograms, but in this case the histograms are used to circumvent the inherent difficulty of segmenting sung melodies. Furthermore, the method for comparing histograms as well as aligning sequences of histograms is quite different in the present work than in [39]. This method is presented in detail in Section 5.

All of the representations described above are time series that require an alignment procedure to yield a robust measure of similarity. This observation forms the underpinning of the present work. In particular, it is found that the dimension of the final representation greatly influence retrieval performance; small representations lend themselves to rapid retrieval whereas large representations lend themselves to robust performance.

One further motivation for this work is the inability to generalize the results of much research in QBH-systems beyond their original scope. In this work we demonstrate how the various systems will perform, relative to each other, as the size of the target database is increased to the size one would expect for a real-world system.

The following subsection provides a brief background and motivations. Methods for evaluation are presented in Section 2. Sections 3, 4, & 5 describe, in turn, the computation and alignment of the contour representation, the note representation, and the histogram representation. The perfor-

mance of the three representations is compared in Section 6.

2 Methodology

One of the many problems confronting the MIR community is the difficulty of comparing results from different projects [9,10]. In particular, different research groups employ different datasets for testing. In this work, rather than present results for a single target database, results will be presented for a range of target databases; the size of the target database is varied, while making sure that the target themes that the test queries represent are always in the target database. As the number of distracting targets in the database increases, the performance of any IR system naturally decreases. In the present work we do not emphasize the performance of our QBH system on any one target database, but rather the rate of decrease in performance as the size of the target database increases. The MIR community is actively developing QBH systems with the hope that such systems may be deployed outside the confines of the research lab. Therefore, it is of the utmost importance to consider how QBH systems perform as the target database is scaled up to massive proportions.

Throughout this work two QBH performance measures will be reported, classification accuracy and mean reciprocal rank. Let r_k be the rank of the correct target for query k. The classification accuracy, CA, is the percentage of recorded queries for which $r_k = 1$. Note that $0 \leq CA \leq 1$. The mean reciprocal rank is given by $MRR = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{r_k}$ where N is the total number of recorded queries used for testing. Note that $CA < MRR \leq \frac{1+CA}{2}$.

2.1 Query Test Set

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 fifteen 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[3]. 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 5 and 20 seconds in length. All data was collected in a quiet classroom setting and participants were free to progress at their own pace. Most of the recorded queries are easily identifiable, if for no other reason than the recordings contained recognizable lyrics. A few dozen of the sung queries are especially poor, however. In particular, some participants had a tendency to sing the melody virtually monotone.

Note that every query represents the exact portion of one of the target themes. For a real-world QBH system, this is an unreasonable assumption. Some systems address this problem by specifically allowing for inserted and deleted notes at the beginning and end of the theme in the retrieval component [29]. Another common practice is to include multiple themes for each tune in the target database [9], increasing the size of the target database.

[3]This contrasts substantially from the common practice of having participants sing isolated pitches on a neutral vowel.

2.2 Target Database

In previous work [2, 3] we employed a target database consisting solely of the fourteen themes for which we have sample queries. While we report classification accuracy in excess of 90% in [2, 3], this measure was not indicative of the the methods would perform in a broader context. Expanding the target database to include themes not represented in the test set was proposed in [9]. In the present work we follow suit and consider retrieval performance as the target database size increases.

How the target database of themes is augmented is of critical importance. The additional themes must be sufficiently similar to the original themes to substantially detriment retrieval performance. That is, augmenting our target database with themes that are very different from those represented in the test set would not necessarily affect retrieval performance. For example, if we augmented our target database with a collection of themes found using a web-crawler, there's no guarantee the web-crawler did not stumble upon a collection John Zorn performances. Scaling the target database to include thousands of similar 'authentic' themes is difficult. Accordingly, we augment the fourteen authentic themes with a varying number of 'synthetic' target themes. It should be noted from the outset that using synthetic targets limits how the results can be interpreted; we are concerned with relative trends rather than absolute performance, however.

We generate synthetic themes with a Markov process designed to yield note sequences with similar first-order statistics as the fourteen authentic themes. For every synthetic target, two parallel Markov processes are constructed. One Markov process generates a sequence of note pitches: each state of the model represents a pitch. And the other Markov process generates a sequence of note durations: each state of the model represents a duration. Note pitches and durations are generated independently.

Two methods were explored for generating the two Markov models. Markov models were implemented that used the Yule algorithm to generate state transition probabilities, using the 14 'authentic' themes as training data. The resultant transition probabilities were unreasonable, however; certain common pitch transitions were not well represented in the small training set. As such, Markov states and transition probabilities were assigned intuitively using the procedure below.

Using the 14 'authentic' targets, histograms are computed for the number of notes per target, the pitch range for each target and the note duration range for each target. These three histograms are normalized to represent probability distributions. For every synthetic target the number of notes as well as the pitch and duration ranges are given by realizations of random variables with the corresponding distributions. The set pitch states is then the set of all uniformly spaced (with 12 pitches per octave) pitches in the allowed range. The duration states are taken as all multiples of two and three times the minimum note duration until the maximum note duration is exceeded. Finally a tonic is chosen at random from among the pitch states.

Having established the pitch and duration states for the Markov models it is necessary to define transition probabilities between states. For a given pitch state, the transition probabilities away from that state are given by a triangular distribution that peaks at the current pitch state and descends linearly to zero at the minimum and maximum allowed pitches. Several modifications to this distribution are then made. First the probability of transitioning to the same state is reduced by a factor of ten and then the probability of transitioning to any state that is not in the key of the selected tonic is reduced by a factor of ten. Transition probabilities are then re-normalized. The duration state transition probabilities are uniform except that the probability of transitioning to the longest durations was reduced by a factor of three and the probability of transitioning to the same

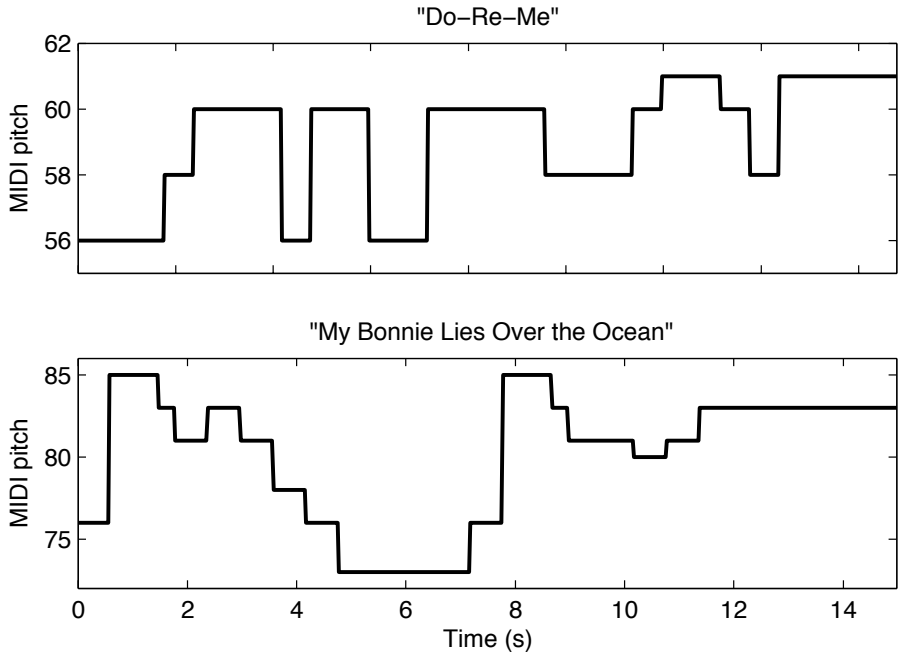

Figure 1: Pitch-contour for two of the tunes in the target database. The top contour is for ”Do-Re-Me” by Richard Rodgers and the bottom contour is for ”My Bonnie Lies Over the Ocean” by Charles E. Pratt.

duration was increased by a factor of two.

The Markov process outlined above yields similar pitch-contours as the fourteen ‘authentic’ themes. Figure 1 shows two ‘authentic’ targets, the top pitch-contour is that of Richard Rodgers’ “Do-Re-Me” and the bottom contour is that of traditional Scottish tune “My Bonnie Lies Over the Ocean.” Notice that there are no breaks in either pitch-contour even though the tunes do contain rests. It has been found that note-offset time is a very unreliable statistic, hence many QBH systems consider the inter-onset interval (IOI) of a note rather than it’s duration [29]. This convention is maintained throughout the present work. Therefore, all notes in a target pitch contour are extended to the beginning to the next note.

Figures 2 & 3 given some sample themes generated by the Markov model. Most of the synthetic targets yielded pitch contours that, at least nominally, resemble the fourteen ‘authentic’ targets, even if they do not sound particularly pleasing (although many do). The Markov process did not always yield reasonable looking pitch-contours however. As can be seen in Figure 3, unreasonable clusters of short notes can be generated. This is due to the Markov constraint: in determining the transition probabilities away from a given state the Markov process cannot consider the path by which the process got to the current state, the transition probabilities only depend on the current state itself. Clusters of short notes are not uncommon in Western music, however such short notes are typically *passing* notes. That is, the general melodic motion is monotonic during sequences of consecutive short notes. This trend is not reflected in the Markov processes implemented in this work.

In spite of the shortcomings outlined above, the Markov process was found to generate syn-

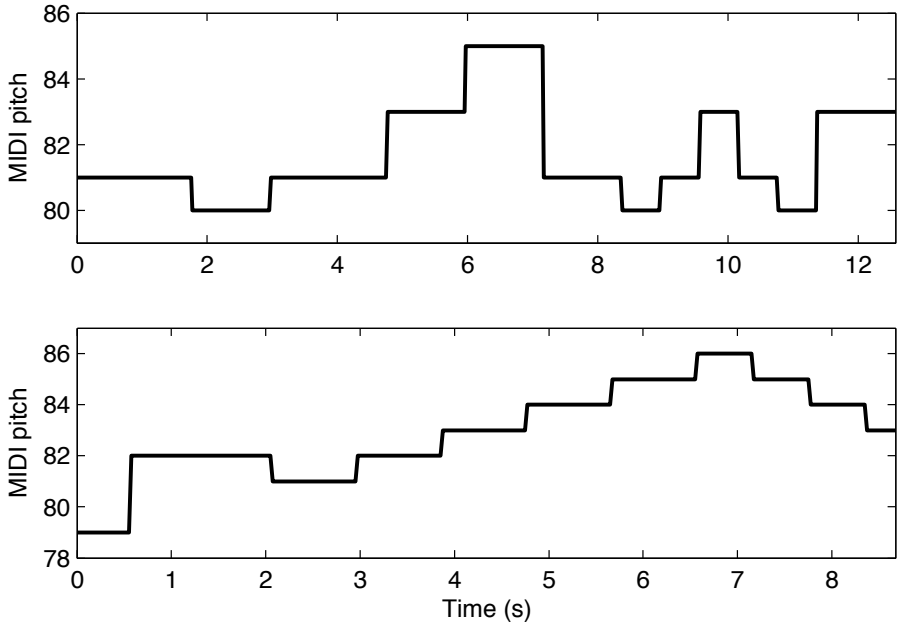

Figure 2: Two reasonable synthetic pitch contours.

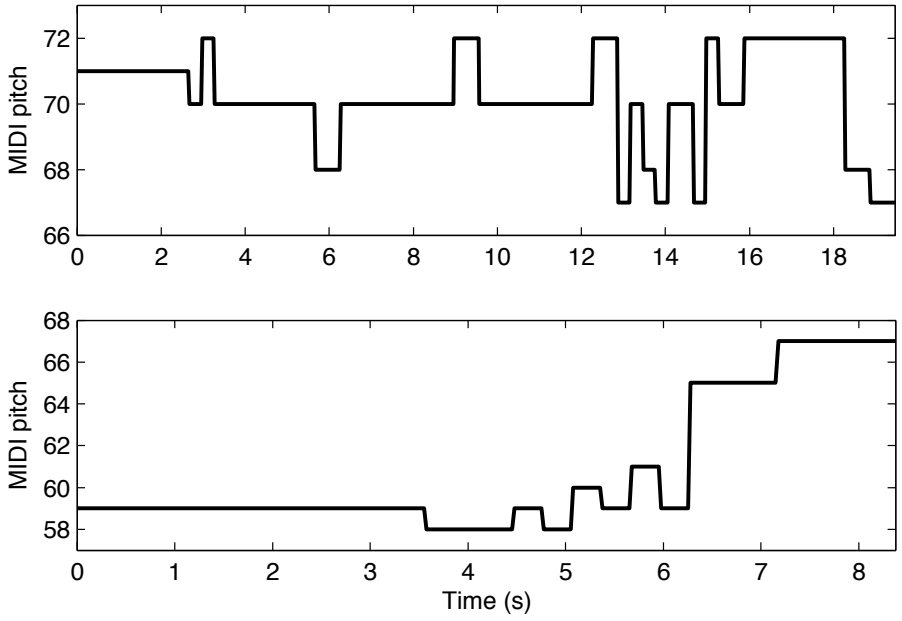

Figure 3: Two unrealistic synthetic pitch contours.

thetic targets sufficiently similar to the fourteen 'authentic' targets to substantially detriment performance. Furthermore, because none of the QBH methods explored in this work consider the global structure of the query, these synthetic targets do not favor one method over the others.

3 Contour Representation

The present work is predicated on the assumption that queries should be coded using primarily pitch information. Singers can sing the same melody with varying amplitude envelopes, lyrics and style. Furthermore, reliable timbrel or phonetic-stream information is difficult to extract from sung melodies [30]. As such, we desire our query representation to be, at least nominally, independent of such variables. An estimate of the sung pitch contour is therefore a natural query representation.

Direct use of the pitch contour was proposed by Mazzoni & Dannenberg [24, 25], in part, to circumvent the difficulty of note segmentation and estimation. While some promising preliminary results were observed in [24,25], subsequent results have been inconclusive [9]. The present work extends some of the ideas presented in [24, 25] and notes some interesting performance trends relative to the other representations discussed below.

The query pitch contour can either be used directly by the retrieval system, or further coded. All three query representations considered in the present work are derived from the pitch contour. The remainder of this section describes the computation and alignment of the contour representation.

3.1 Pitch Contour

Both [25] and the present author use an autocorrelation-based algorithm for estimating the pitch contour, but the two methods 'tweak' the autocorrelation in different ways. Mazzoni & Dannenberg use an algorithm proposed by Tolonen & Karjalainen [40] in which the autocorrelation is lag-scaled and subtracted from itself to remove false peaks[4]. Smooth contours are then constructed using a nearest-neighbor algorithm.

In the present work we use an algorithm proposed by Boersma [4, 6], which we found to be robust to the volatile pitch fluctuations common to untrained singers. The algorithm computes the autocorrelation for overlapping windows of recorded data. The bias of the window function is mitigated by the normalization $\tilde{r}(\tau) = \frac{r(\tau)}{r_w(\tau)}$, where $r(\tau)$ is the autocorrelation of the windowed data and $r_w(\tau)$ is the autocorrelation of the window function. A set of candidate peaks is selected for every frame and the Viterbi algorithm is used to construct a smooth contour. We use a step-size of 10 ms throughout this work. The fundamental frequency values are then converted to MIDI pitch[5], yielding the final pitch contour.

Fig. 4 shows a sample pitch contour for the main theme from "Sound of Music." Included in Fig. 4 is the piecewise constant ideal contour for the same theme stored in our target database. We note that most sample queries in test set yield 'messier' contours, with prolonged scooping between notes as well as considerable fluctuation within notes. In Fig. 4, the target contour has been time-scaled to the same length as the query. Finally, note that the query contour contains gaps corresponding to short pauses taken by the singer; the pitch track algorithm performs an implicit voiced/ unvoiced segmentation.

[4] A signal with period T_0 demonstrates peaks in its autocorrelation at lags $T_0, 2T_0, 3T_0 \cdots$.

[5] The real-valued MIDI pitch number p is related to a signal's fundamental frequency in Hz, f_0, as $p =$

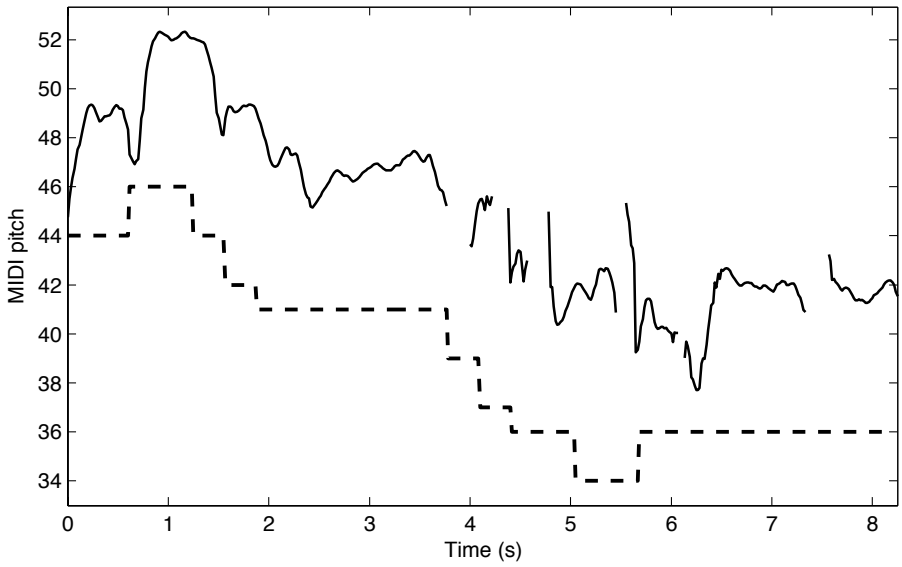

Figure 4: Sample query and target pitch contours for the main theme from Richard Rodgers' "Sound of Music." The query pitch contour is given by the solid line and the target pitch contour is given by the dashed line.

3.2 Dimension Reduction

Direct use of the pitch contour yields a query representation of relatively large dimension. For a step-size of 10 ms, typical pitch contours are of length $\sim 10^3$, whereas the note sequences discussed in the next section are of length $\sim 10^1$. Direct use of the contour is too computationally burdensome. Accordingly, we employ a simple dimension reduction technique proposed in [18]; query pitch contours are decimated by a factor of 10, yielding a representation of length $\sim 10^2$ [24]. Decimating by factors greater than ten effectively smear notes together[6] while factors smaller than ten yield contours that are too computationally cumbersome to align.

A similar question is addressed in section 3.5. We propose the use of a piecewise linear approximation to the pitch contour. By varying the segment approximation error, we can indirectly adjust the dimension of the query representation; trading performance for computation speed.

3.3 Dynamic Time Warping

The retrieval component compares the query pitch contour with the piecewise constant contour for every theme stored in the target database. Let the query pitch contour be given by $\mathbf{Q} = (q_1, q_2, \cdots q_K)$ and the target contour given by $\mathbf{T} = (t_1, t_2, \cdots t_N)$. A direct, albeit 'brittle,' dissimilarity measure is given by [18]

$$D = \sum_{k=1}^{\min(K,N)} |q_k - t_k|^p. \tag{1}$$

$12 \log_2(f_0/261) + 60$.

[6] 'short notes' can be less than 200ms in duration.

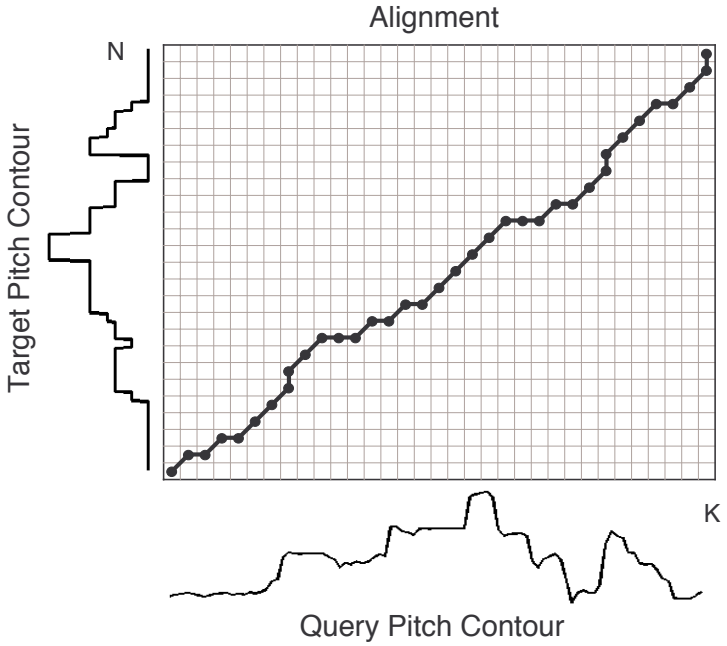

Figure 5: Sample alignment for a query contour of "Row, Row, Row Your Boat" with the true melody.

A more robust metric is yielded with dynamic time warping (DTW), an alignment procedure popularized in the 1970's for speech recognition [14, 36, 37]. The increasing number of multimedia applications for large collections of time series has recently reinvigorated interest in DTW within the database community [18, 34, 42, 43].

To find the optimal alignment path between $\mathbf{Q}$ and $\mathbf{T}$ we recursively compute an $N \times K$ matrix of minimum prefix alignment costs, $\mathbf{\Gamma} = [\gamma_{n,k}]$. $\gamma_{n,k}$ is the minimum alignment cost for $(q_1 \cdots q_k)$ and $(t_1 \cdots t_n)$. Let a warping path be given by $\mathbf{w} = (w_1, w_2, \cdots w_T)$, where $w_t = (n, k)$ indicates that q_k is aligned with t_n. Figure 5 shows a warping path for a sample query pitch contour for the main theme from "Row, Row, Row Your Boat" with the piecewise constant contour of the true theme.

3.3.1 Continuity Constraints & Cost Schemes

The warping path must adhere to several constraints to be physically meaningful. The path must originate in the lower-right corner of $\mathbf{\Gamma}$ and terminate in the upper-left corner; $w_1 = (1, 1)$ and $w_T = (N, K)$. The path must also be monotonically nondecreasing and continuous in some sense. Continuity constraints are often employed to restrict the slope of the alignment path and prevent pathological alignments [14, 34, 36, 37, 42]. In the present work we consider three common continuity constraints; the general constraint [24, 36], the 'Itakura' constraint [14] and the 'Sakoe' constraint [37]. The Itakura and Sakoe continuity constraints place bounds on the slope of the alignment path. The three continuity constraints are shown in Fig. 6. Starting in the lower left

corner of $\mathbf{\Gamma}$, every element of $\mathbf{\Gamma}$ is found recursively by

$$\gamma_{n,k} = \min \begin{pmatrix} \gamma_{n-1,k} \\ \gamma_{n,k-1} \\ \gamma_{n-1,k-1} \end{pmatrix} + \text{Match Cost}(q_k, t_n) \tag{2}$$

for the general continuity constraint, where

$$\text{Match Cost}(q_k, t_n) = |q_k - t_n|^p. \tag{3}$$

The precise value of p has little influence on performance, $p = \frac{1}{2}$ appears to be a good value[7]. For the general continuity constraint, it is also possible to use an edit-distance cost scheme,

$$\gamma_{n,k} = \min \begin{pmatrix} \gamma_{n-1,k} & + & 1 \\ \gamma_{n,k-1} & + & 1 \\ \gamma_{n-1,k-1} & + & \frac{2}{\alpha}\,\text{Match Cost}(q_k, t_n) \end{pmatrix}. \tag{4}$$

For the edit-distance cost scheme, vertical and horizontal steps in the warping path are interpreted as inserting and deleting elements from the query. Insertion and deletion steps are assigned a constant cost and diagonal steps are assigned a cost proportional to the match cost between the two pitches. For $p = 1$, matching two pitches with a difference of α yields a cost equivalent to deleting the old pitch and inserting the new pitch, similar to the cost scheme described in section 4.3.1. For the Itakura and Sakoe constraints, the recursive cost equations are

$$\gamma_{n,k} = \min \begin{pmatrix} \gamma_{n-2,k-1} & & \\ \gamma_{n-1,k-1} & & \\ \gamma_{n-2,k-2} & + & \text{Match Cost}(q_{k-1}, t_n) \\ \gamma_{n-1,k-1} & + & \text{Match Cost}(q_{k-1}, t_n) \end{pmatrix} + \text{Match Cost}(q_k, t_n) \tag{5}$$

and

$$\gamma_{n,k} = \min \begin{pmatrix} \gamma_{n-1,k-1} & & & & \\ \gamma_{n-2,k-1} & + & \beta\,\text{Match Cost}(q_k, t_{n-1}) & & \\ \gamma_{n-1,k-2} & + & \beta\,\text{Match Cost}(q_{k-1}, t_n) & & \\ \gamma_{n-3,k-1} & + & \beta^2\,\text{Match Cost}(q_k, t_{n-2}) & + & \beta\,\text{Match Cost}(q_k, t_{n-1}) \\ \gamma_{n-1,k-3} & + & \beta^2\,\text{Match Cost}(q_{k-2}, t_n) & + & \beta\,\text{Match Cost}(q_{k-1}, t_n) \end{pmatrix} + \text{Match Cost}(q_k, t_n), \tag{6}$$

respectively. $\beta \geq 1$ is an extra cost penalty applied to favor more direct alignment paths. We found $\beta \approx 1.4$ to be a good value.

The final alignment cost for $\mathbf{Q}$ and $\mathbf{T}$ is given by $\gamma_{N,K}$, and is interpreted as a dissimilarity measure. By performing this alignment on every target in the database we can rank order the target themes. The complexity of this alignment procedure is $O(NK)$. Note that the final alignment cost is *not* normalized by the total length of the warping path. Using $\gamma_{N,K}/T$ as the final similarity metric is common in DTW systems [24, 36, 37]. This normalization is used so as to not penalize long targets. However, as discussed below, in our system all targets are time scaled to have duration

[7]The current implementation also uses the remainder of dividing $|q_k - t_n|$ by 12 so as to make the match cost invariant to octave errors.

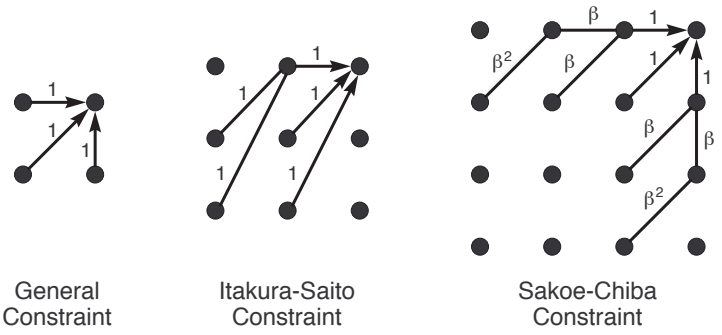

Figure 6: Three local continuity constraints for DTW. The Sakoe-Chiba continuity constraint is shown with a slope constraint of $\frac{1}{2}$ [37].

equal to that of the query. Normalizing by the total warping path length prohibits the use of many cost schemes, as the DP algorithm is no longer guaranteed to find the optimal alignment path after normalization. The cost scheme we use for the Sakoe continuity constraint, for example, violates the DP constraint if the final alignment costs are normalized. As such, we do not normalize the final alignment costs.

Note that whether the use of a cost scheme or continuity constraint that violates the DP constraint (in which case the final alignment path is suboptimal) detriments *retrieval* performance is an unanswered question. Indeed, other cost schemes were explored that violate the DP assumption even without normalization. One of these cost schemes yielded marginally better performance than even the best cost scheme described above (although this improvement is virtually negligible). When using such a cost scheme, the alignment procedure finds a suboptimal alignment for all target themes, not just the correct theme. So long as the alignment procedure does not 'penalize' the correct theme more than the others, it does not really matter whether or not an optimal alignment is found.

3.3.2 Implementation Issues

The following implementation issues were found to have a significant effect on system performance, but are not particularly interesting to discuss. The target database used throughout this work stores targets as a sequence of (pitch, duration) pairs. Most QBH systems must be tempo-invariant: only the relative durations in the target database are important, not the absolute values. The target pitch contours time-scaled to be the same length as the query contour.

In addition to being tempo-invariance, most QBH systems must also be transposition-invariant. The average pitch offset between the query and target is subtracted prior to alignment. We can iterate between these two steps, pitch-shift and alignment, to further improve the match quality. This two-step iteration was implemented and preliminary results showed that while the initial pitch-shift is essential, successive iterations did not improve *retrieval* performance. The DTW algorithm is computationally expensive even without this iteration and hence this iteration is not included in the performance results below.

The pitch-contour computed for the query is not continuous: when the singer pauses or the pitch is otherwise impossible to estimate, the pitch-track algorithm outputs a 'null' pitch, as seen

in Fig. 4. In fact, the pitch-contour itself is a voiced/unvoiced detection. It is unclear how to match such 'null' query regions to the target contour. Multiple cost schemes were implemented for the null regions, but simply 'filling in' the null regions was found to yield the best performance. That is, the end of every segment of detected pitch is extended to the beginning of the next segment. This extension is performed by computing the weighted average of the last 100ms of each segment and setting the entire null region to this value.

It should be evident from Figure 5 that allowing extreme warping functions is not useful. That is, allowing the alignment path to stray into the upper-left or lower-right corner of $\mathbf{\Gamma}$ is unrealistic. Two contours that are reasonably similar in shape should not require such distorted alignments. Restricting the alignment path to a certain width, referred to as the beam-width, not only prevents pathological alignments but also increases the speed of computation. In this work we found using a beam-width of 20% of the query length to work well.

3.4 Performance

Fig. 7 displays the performance of the five QBH systems described above, all of which use a pitch contour representation. For both plots in Fig. 7, the target database size is represented along the abscissa. The ordinate represents classification accuracy for the left plot and MRR for the right plot. The fourteen 'authentic' themes are included in every database size, hence for the largest target database size, 3570 of the 3584 themes are 'synthetic.' For all but the smallest and largest target databases, the points shown are an average across multiple target databases. Data are shown along with best-fitting linear curves. This is in contrast to [9], where performance was found to be inversely proportional to the log of the target database size. Clearly, the linear fit cannot be extrapolated indefinitely, CA cannot become negative. Nonetheless, a linear fit provides a pragmatic visual aid and implies a 'slope,' or rate of performance degradation.

All methods show similar performance for the smallest target database, with classification accuracy between 85% and 90%. As the target database grows however, some clear differences in performance become evident. For the largest target database classification accuracy in between 55% and 80%. The curve labelled "Direct" gives the performance of a QBH system that uses (1) to compare query and target contours, without alignment[8]. As expected, this rudimentary dissimilarity measure yields the worst performance. The standard DTW cost scheme, given by (2), yields substantially better performance, but the other cost schemes yield better performance still.

The edit-distance, Itakura and Sakoe cost schemes yield similar performance, with $CA \approx 89$ for a target database with 14 themes and $CA \approx 79$ for a target database with 3584 themes. The Itakura and Sakoe cost schemes yield slightly better performance than the edit-distance cost scheme. In particular, the Sakoe constraint yields the most robust performance; the rate of performance degradation for this cost scheme is slower than the rest, even if it performs slightly worse than the Itakura cost scheme for small database sizes.

While the improved Sakoe and Itakura constraints yield somewhat better performance than the edit-distance constraint, there may be a subtle advantage to using the edit-distance constraint. The subset of the matrix $\mathbf{\Gamma}$ that the alignment is allowed to visit is known as the *warping window* [17]. Figure 8 shows the warping window for the edit-distance and Itakura constraints. Dark squares

[8]Note that unlike the other performance curves shown in Fig. 7, the 'Direct' curve is for a system that does not down-sample the contours as described in section 3.2. If the contours are down-sampled prior to computing (1), performance is considerably worse than that shown in Fig. 7.

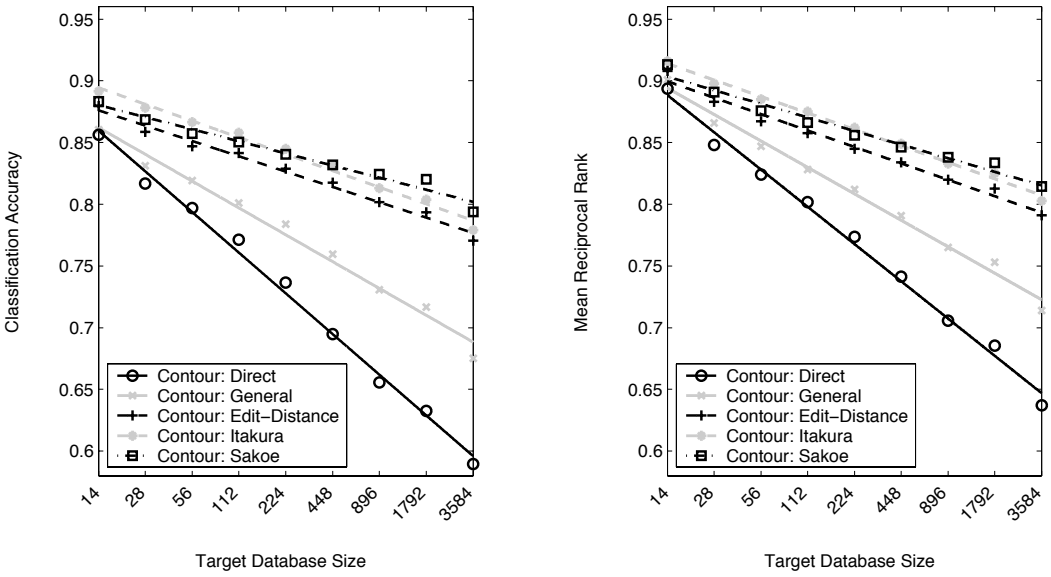

Figure 7: Performance of five QBH systems that use a pitch contour representation. The left plot gives the classification accuracy versus target database size and the right plot gives the MRR versus target database size.

show the points the alignment path can visit. One of the global constraints for the alignment is that it must begin at $\gamma_{1,1}$ and end at $\gamma_{N,K}$. The edit-distance constraint allows the beginning and end of the query to be inserted or deleted to match the target contour. The Itakura and Sakoe constraints force the beginning and end to expand or contract only 'gradually' however [17]. For example, if the query contains a portion at the beginning that does not match with the beginning to the target contour, but the contours are similar otherwise, the edit-distance constraint allows the spurious portion at the beginning to be deleted (resulting in a small alignment cost) whereas the Sakoe and Itakura constraints do not allow the spurious portion to be completely contracted (resulting in a large alignment cost). As mentioned in Section 2.1, real-world QBH systems will be expected to operate even when the user does not sing a theme using the same start and end notes as the theme stored in the target database. The edit-distance constraint may be better suited to this environment than the Sakoe and Itakura constraint. On the other hand, the boundary conditions given above can be loosened, allowing the deleted or spurious portions at the beginning and end of the query to be accounted for while using a continuity constraint that restricts the slope of the warping path [29].

3.5 Piecewise Linear Approximation

Section 3.2 describes the role of down-sampling in reducing the dimension of the query contour to a manageable size. Down-sampling can be viewed as approximating the signal as piecewise constant [18]. The constant regions are of equal length, in this work length ten. Another way to approximate the contour is as piecewise linear, where segment boundaries are defined using a maximum allowable segment approximation error rather than a constant segment length. This approach to dimension reduction has been recently explored by Keogh et al [8, 19–21]. Of immediate

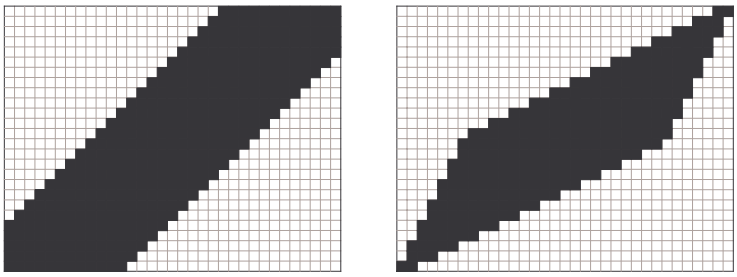

Figure 8: Alignment path boundaries for the general and Itakura continuity constraints.

applicability to the current work is Keogh's use of DTW to measure the similarity between time series approximated by such piecewise linear representations [20].

3.5.1 Implementation

In the present work the piecewise linear approximation is computed using a simple 'bottom-up' recursive algorithm [19, 21, 33]. The initial approximation is given by consecutive pairs of contour values; there are $\frac{K}{2}$ initial segments yielding zero approximation error. The algorithm then merges the two neighboring segments that result in the smallest increase in approximation error (found by integrating the absolute difference between the original contour and linear approximation). Linear regression is used to find the best linear fit for every segment of observed contour. The algorithm terminates when the lowest cost merge increases the individual segment error above some threshold, ε. Figure 9 shows two sample piecewise linear approximations for a portion of the main theme from "America the Beautiful." The top plot shows the approximation made for $\varepsilon = 2$ and the bottom plot shows the approximation made for $\varepsilon = 16$. This algorithm is readily modified to yield piecewise constant approximations as well.

The DTW algorithm using the simplest continuity constraint (Equation 2) is used for the piecewise linear representation, although a modified cost scheme is required. Let the piecewise linear approximations of $\mathbf{Q}$ and $\mathbf{T}$ be given by $\hat{\mathbf{Q}} = (\hat{\mathbf{q}}_1, \hat{\mathbf{q}}_2, \cdots \hat{\mathbf{q}}_{\hat{K}})$ and $\hat{\mathbf{T}} = (\hat{\mathbf{t}}_1, \hat{\mathbf{t}}_2, \cdots \hat{\mathbf{t}}_{\hat{N}})$. For the contour representation described above, each sequence element is simply a pitch. For the note representation described in the next section, each sequence element is a (pitch, duration) pair. For the current piecewise linear representation each element is a (mean pitch, pitch slope, duration) triple, $\hat{\mathbf{q}}_k = (\hat{q}_{k,\,\text{pit}},\ \hat{q}_{k,\,\text{slope}},\ \hat{q}_{k,\,\text{dur}})$. In [20] Keogh proposes the following match cost,

$$\text{Match Cost}(\hat{\mathbf{q}}_k, \hat{\mathbf{t}}_n) = |\hat{q}_{k,\,\text{pit}} - \hat{t}_{n,\,\text{pit}}|^p \tag{7}$$

Keogh set $p = 2$ but in for the present application we found $p = \frac{1}{2}$ to be a better value. Given that this match cost makes no use of the slope or duration of either segment it is unclear why Keogh proposes a piecewise linear approximation instead of a piecewise constant approximation. In this work we also implemented a piecewise constant representation using the same cost scheme and found equivalent performance.

A more intuitive match cost might be to integrate the difference in area between the linear segments. It is unclear how to perform this 'integration' however because the two segments are not

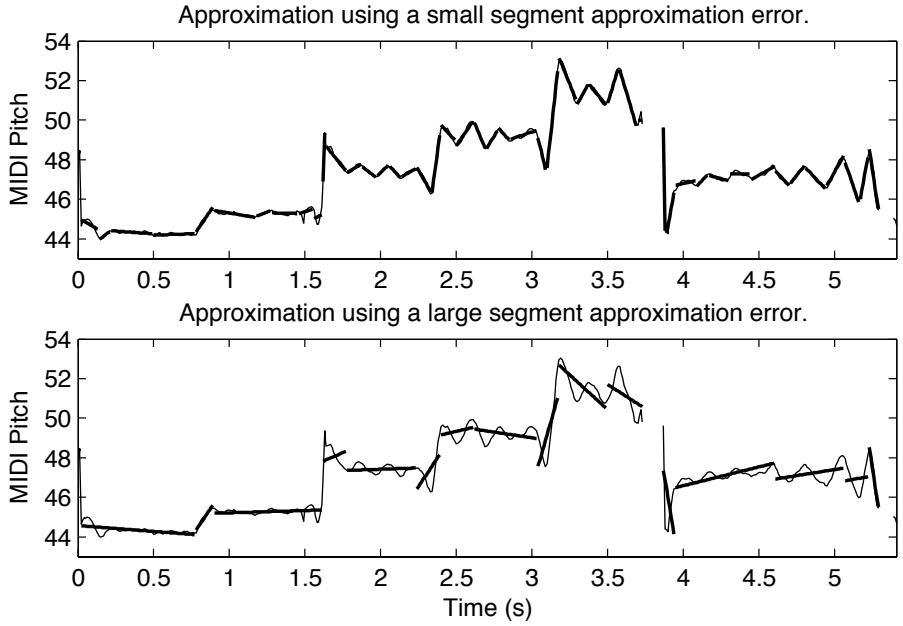

Figure 9: Sample query pitch-contour for a portion of the main theme to "America, the Beautiful." The thin line is the observed pitch-contour and the thick line is a piecewise linear approximation. The top plot shows a piecewise linear fit with small approximation error and bottom plot shows a fit with large approximation error.

necessarily the same length. In the present work we consider the following heuristic approach. The piecewise linear approximation to the target contour clearly contains a small number of constant segments; $\forall\, n,\ \hat{t}_{n,\,\text{slope}} = 0$. Hence a reasonable match cost is

$$\text{Match Cost}(\hat{\mathbf{q}}_k, \hat{\mathbf{t}}_n) = \sum_{i=1}^{q_{k,\,dur}} |(q_{k,\,pit} + i \cdot\ q_{k,\,slope}) - t_{n,\,pit}|^p \tag{8}$$

While this match cost would appear to better capture the difference between two segments, it was found experimentally to perform somewhat worse than the simpler match cost above. This is likely a result of the disparity between query and target sequence lengths, as discussed below.

3.5.2 Performance

The performance of the piecewise linear system for various levels of approximation error ε is shown in Figure 10. As expected, the performance of the system improves as the approximation error decreases (and the average dimension increases). It is also clear from the figure that the piecewise linear system performs considerably worse than even the 'brittle' direct method, regardless of how small the approximation error is.

There are several possible reasons for the poor performance. One shortcoming of the current implementation is that the piecewise linear approximation algorithm returns a much longer sequence of segments for the query contour than the target contour. The target contours, which are

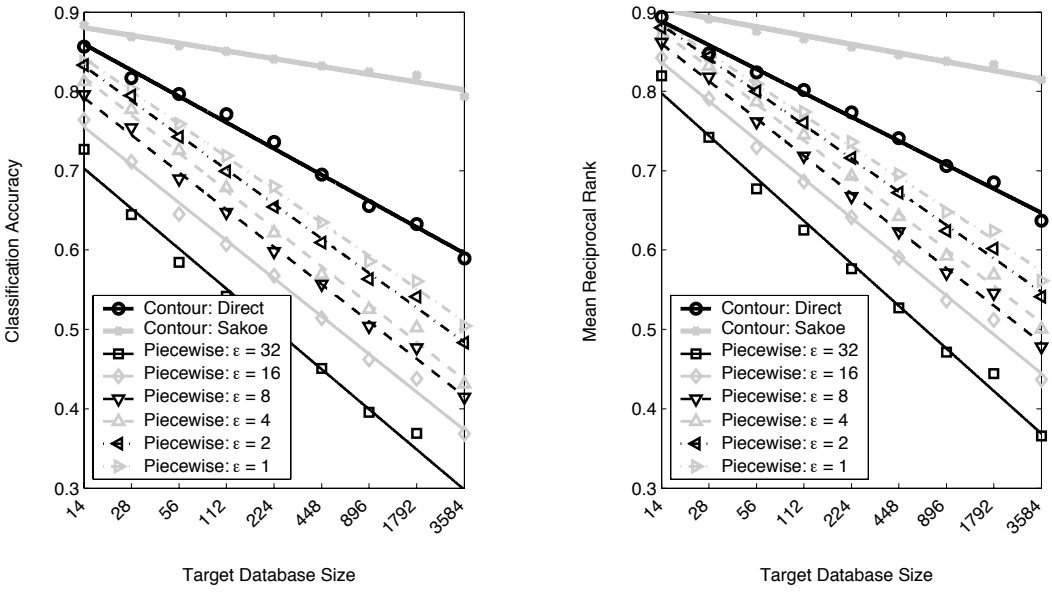

Figure 10: Performance of the piecewise linear contour representation.

piecewise constant, yield a piecewise linear representation containing as many segments as there are notes in the melody; typically $\hat{N}$ is between ten and twenty. For the query contours however the piecewise linear representation contains between 30 and 100 segments. Because $\hat{K}$ and $\hat{N}$ are so disparate it is not possible to apply a continuity constraint that biases the slope of the alignment, as for the Itakura and Sakoe constraints. The consistent difference between $\hat{K}$ and $\hat{N}$ could also explain why the match cost (7) yields better performance than the more intuitive match cost (8). This disparity could be addressed by simply segmenting the target representation so as to have as many segments as the query; perhaps by inserting into $\hat{\mathbf{T}}$ segment boundaries wherever there are segment boundaries in $\hat{\mathbf{Q}}$.

While the current piecewise linear implementation is not effective in terms of absolute retrieval performance, it does facilitate further investigation into the influence of representation dimension on retrieval performance. In Figure 10 the different piecewise linear curves are labelled with the *segment* approximation error of the piecewise linear representation. The curves can also be labelled using either the average total approximation error or the average representation length. These two cases are shown in Figure 11. The ordinate gives the MRR, the depth axis gives the target database size and the horizontal axis gives the total approximation error for the left plot and the average representation length for the right plot. As is evident from the figure, the performance is monotonically increasing with respect to representation length for all target database sizes.

Figure 12 averages the MRR across target database size for the surface shown in Fig. 11. For both plots in Fig. 12 the ordinate gives the MRR. For the left plot the abscissa gives the total approximation error and the right plot gives the representation length. The difference in shape demonstrated in Fig. 12 is worth noting. The MRR rank appears to to degrade linearly as a function of the approximation error. A linear fit is clearly not appropriate for MRR versus representation length however; performance appears to plateau for lengths greater than 100. Performance improves much more when the representation length is increased from 20 to 30 than from 80 to 90

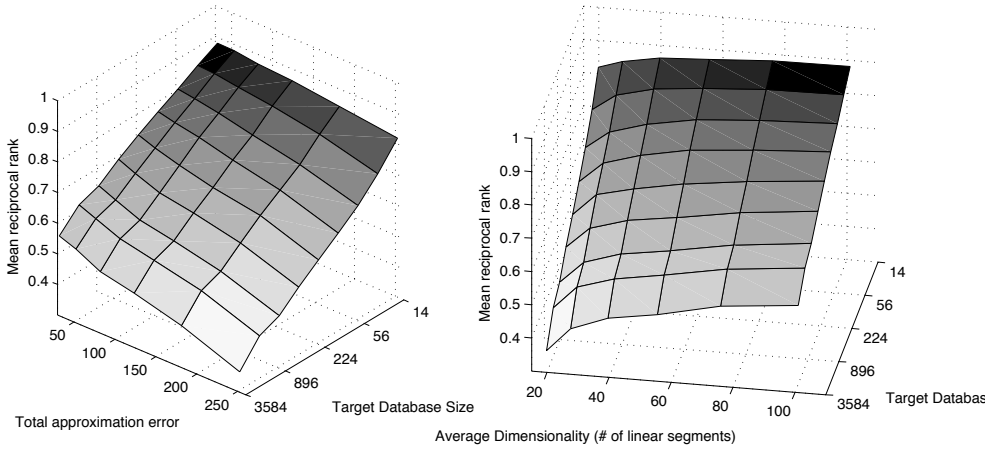

Figure 11: Mean reciprocal rank for the piecewise linear contour representation. The left plot gives the MRR versus total approximation error and the right plot gives MRR versus average query dimension. The depth axis gives the size of the target database.

(or even 60 to 90). This would seem to imply that a representation 'finer' than notes is preferable for QBH systems. Nonetheless, note representation are standard in practice. The next section discusses note representations.

4 Note Representation

Most QBH systems operate by first transcribing the sung query into a sequence of (pitch, duration) pairs [3, 9, 11, 27, 29, 32]. In addition to being musically intuitive, melody transcription is one method for reducing the dimension of the query for database searching. The present work compares five note transcription methods. Three of the note estimators have been previously described [1,2]. The other two new methods will be described in greater detail [3].

4.1 Note Segmentation

Often, naturally sung melody transcription is implemented as a three-stage process. First, the sung pitch contour is estimated. This contour is then segmented into separate notes. Finally, note pitches are assigned as the average contour value within each note. Performing note segmentation prior to pitch assignment is intuitive; it is unclear how to assign pitches to individual notes before note boundaries have been fixed. Simultaneously searching the space of all possible note boundaries and note pitches is considerably more difficult than first searching the space of note boundaries and then assigning pitches [1, 16].

Many QBH systems [11, 22, 23, 28] require the user to articulate each note with a separate 'da' or 'ta'; the user is required to perform note segmentation. In this case, a simple amplitude threshold is used to detect note boundaries. If a user sings a continuous melodic line or lyrics the segmentation process fails and the QBH systems yield poor performance. It has been shown that even when the user adheres to the above restriction, the majority of errors result from this

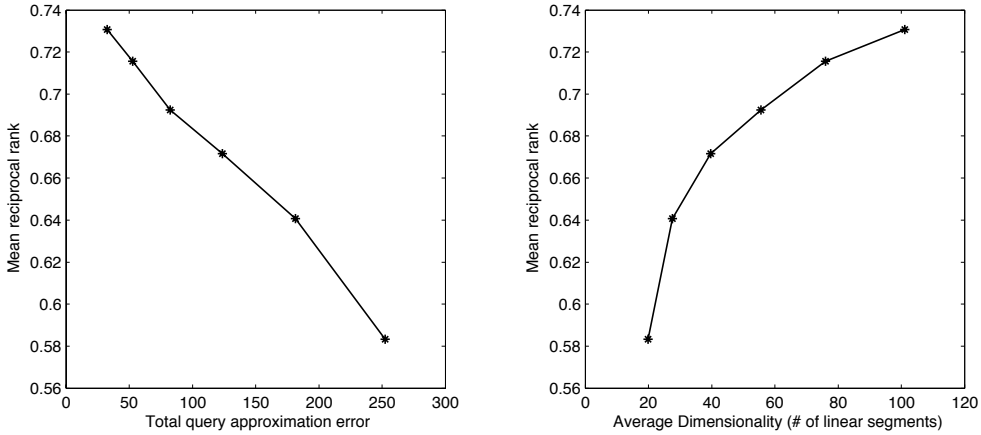

Figure 12: Mean reciprocal rank for a piecewise linear contour representation. The plot on the left gives the performance vs. total approximation error and the right plot gives the performance vs. average query dimension.

rudimentary note segmentation method [26,29]. This restriction is undesirable however, especially for systems designed for untrained singers.

Three segmentation methods presented in [1] are included in the present work: a smoothed pitch derivative [28], an adaptive RLS filter, and a nonlinear LMS filter (or perceptron). Two other methods that incorporate a prior constraint on the distribution of note pitches are also included: a quantizer and an HMM [3]. All five of the note estimators included here employ the pitch detection algorithm described in section 3.1. The various note estimators are included to demonstrate the range of performance possible using a note representation.

4.2 12-Tone Prior

The note segmenters included in [1] do not make use of the fact that most melodies found in Western music are restricted to a 12-tone scale. While incorporating a 12-tone, equal-tempered, prior into note segmentations was not found to improve segmentation performance, it was found to improve retrieval performance considerably [2,3].

There are a few concerns however with using this prior that often discourage its use. The first difficulty is simply that most individuals do not have perfect pitch, and set their own personal tonic when singing a solo melody. The 12-tone equal-tempered prior only constrains the distribution of pitches, not the absolute pitches themselves. A simple resolution to this difficulty is presented below.

Another concern with applying the 12-tone equal-tempered prior is pitch drift. Singers, especially untrained singers, will slowly modify the tonic pitch throughout a solo performance. While pitch drift presents a considerable challenge to general sung melody transcription, it was found that for the queries used in this work pitch drift is negligible. Sung queries for QBH retrieval are generally less than 20 seconds in duration, in which case even an untrained singer will not drift too far from their original tonic.

In this work we incorporate a 12-tone, equal-tempered, prior into two note segmenters. Both

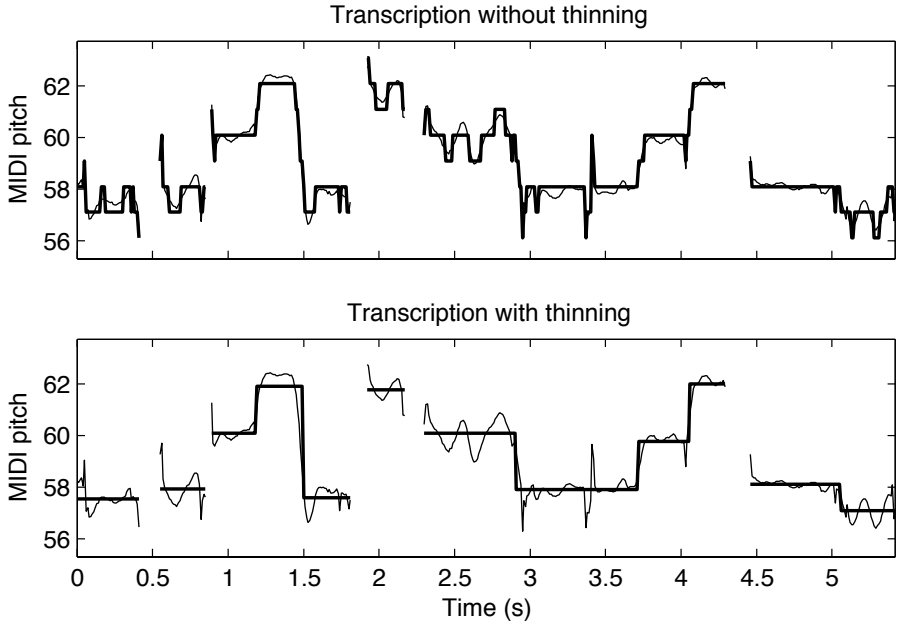

Figure 13: Transcriptions for a sample query of the main theme from "Yankee Doodle." The thin line is the observed pitch-contour and the thick line is the transcribed contour. Both transcriptions were generated using the quantizer segmenter. The top plot shows the result when no note thinning is performed, the bottom plot shows the case when note thinning is performed.

segmenters require that a codebook of quantization pitches be specified before segmentation. An initial codebook spanning several octaves is constructed with 12 quantization levels per octave. For a set of pitch offsets spanning one half-step[9] separated by 5 cents[10]100 cents equal a half-step.), the quantization MSE is computed for the observed pitch contour. The offset that yields the smallest quantization error is chosen for the final codebook design.

4.2.1 Quantizer as estimator

A simple method for detecting note boundaries is to use the quantizer described above to indicate note boundaries; a note is inserted wherever the quantized contour transitions between pitches. As expected, such a segmentation is riddled with spurious notes [3,34]. The top panel of Fig. 13 shows the direct quantizer output for a sample query of "Yankee Doodle." Spurious notes are a problem in most transcription systems [3, 28] and are often dealt with by merging all notes less than some duration, T_{min}, with their nearest neighbor.

Applying minimum duration thinning to the quantized contour dramatically reduces the number of spurious notes. The algorithm iteratively selects the shortest note and *merges* it until the shortest note in the quantized sequence is greater than T_{min}. We found $T_{min} \approx 150$ms to be a

[9] An interval of 1.0 in MIDI pitch.

[10] (

good value.

In spite of the minimum duration thinning, numerous spurious notes persist. To further reduce the number of spurious notes a minimum pitch difference constraint is applied. For each note, a preliminary pitch is assigned as the average *unquantized* contour value within the note's boundaries. The difference in pitch between adjacent pairs of notes is computed. The algorithm iteratively selects the smallest pith difference and merges the two notes together until the smallest pitch difference is greater than ΔP_{min}. We found $\Delta P_{min} \approx 0.85$ to be a good value [3].

The bottom panel of Figure 13 shows the result of this process applied to the quantized contour. The two thinning procedures described have, in this case, removed all the spurious notes without removing any true notes. Removing the 'minimum pitch' thinning procedure would introduces a small number of spurious notes back into the estimated note sequence; the minimum duration constraint is the more important of the two.

4.2.2 Hidden Markov Model

Another technique for incorporating the 12-tone, equal-tempered, prior that is more robust to spurious notes is an HMM. Every state in the HMM represents one unique pitch in the 12-tone equal-tempered codebook described above [2, 3]. Spurious notes are limited by defining state transition probabilities that favor constant pitch.

Two probability distributions were implemented. A rudimentary distribution that sets the probability of self-transition very high (about 98%) and uniform for the remaining pitch states ($\frac{2\%}{M-1}$ where M is the number of HMM states). Another transition probability distribution was computed with the Yule algorithm, using the fourteen 'authentic' target themes are training data [35]. Surprisingly, the rudimentary transition probability distribution was found to yield considerably better performance, perhaps due to the small set of melodies used for training.

In [1] it was found that if the observed pitch-contour is modelled as piecewise constant with additive noise, a Laplacian distribution is a good model for the noise process. Accordingly, the observation noise process used by the HMM is Laplacian with $\beta \approx 3$.

Having constructed the HMM, the most probable state sequence is found with the Viterbi algorithm, using the pitch contour as the observation sequence [35]. This algorithm constructs a trellis to record the most probable prefix path to every state for every time step of observation data. The algorithm then regresses back through the trellis to find the most probable note sequence.

An example transcription using the HMM is shown in Fig. 14for the same sample query as in Fig. 13. By comparing the two figures it should be clear that the Quantizer and HMM produce similar transcriptions. It is worth noting that the HMM estimator does not require a minimum duration thinning step like most melody transcription methods. This is because the high self-transition probability for each state naturally discourages spurious notes. It was found that the minimum duration thinning did improve performance modestly however, so the step was included in final performance comparisons. Lastly, while the two methods often yield similar transcriptions, the quantizer is considerably faster than the HMM. While the Viterbi algorithm keeps the computational complexity of finding the optimal state path manageable, it is still much slower than quantizer described above.

4.3 String Matching

The query representation is thus given by a sequence of (pitch, duration) pairs. The query sequence must be compared with every target sequence in the database. Due in part the the difficulty

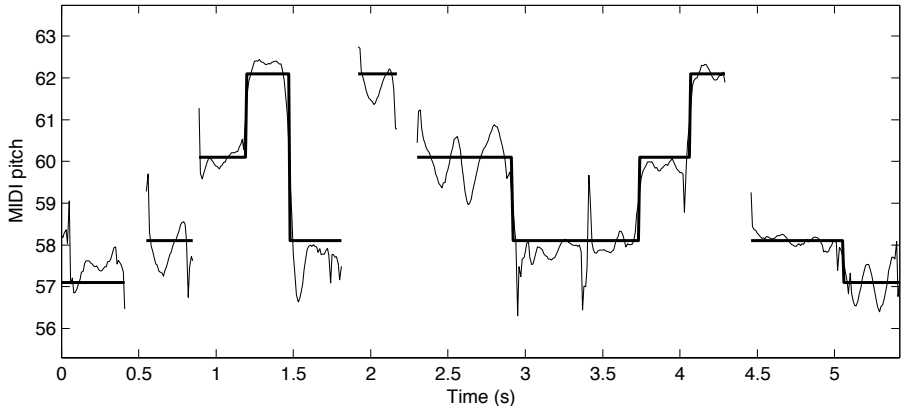

Figure 14: Transcription for a sample query of the main theme from "Yankee Doodle." The thin line is the observed pitch-contour and the thick line is the transcribed contour, generated using the HMM method.

of reliably detecting note boundaries, a direct comparison of query and target sequences yields poor performance. A more robust method for comparison is taken from biological sequence analysis [12].

Let the estimated melody be given by a sequence of K 'notes' pairs, $\mathbf{Q} = (\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_K)$ where $\mathbf{q}_k = (\, q_{k,\text{pit}}\,,\, q_{k,\text{dur}}\,)$. Similarly, let a target theme be given by a sequence of N 'notes' pairs, $\mathbf{T} = (\mathbf{t}_1, \mathbf{t}_2, \cdots, \mathbf{t}_N)$. Because note off-set time is an unreliable statistic, inter-onset interval (IOI) is used in the place of note duration [29].

The prevailing method for aligning two sequences of notes is string matching [12]. The alignment is achieved by inserting, deleting, and replacing elements of the query sequence in order to match the target sequence. Each of these three operations is assigned some cost, and a dynamic algorithm is employed to find the minimum cost alignment.

4.3.1 Cost Scheme

Numerous cost schemes have been proposed, and we found a simple approximation to an edit-distance DTW metric to be particularly effective Suppose both the query and target note sequences are 'unwrapped' into the piecewise constant pitch contours they represent. A unit cost is assigned to the insertion or deletion of a sample from the query contour. Further, we define the cost of replacement to be proportional to the pitch difference between two samples. In this formulation, there will be some pitch difference above which replacement will cost more than simply deleting the note and inserting a new sample. We denote this pitch difference as α.

This cost scheme cannot be implemented directly on the note sequences: 'unwrapping' the note sequences for alignment allows the optimal alignment to split query notes between target notes. When aligning the note sequences, the notes must be inserted or deleted in their entirety. The cost scheme outlined above can be translated to string edits on complete notes however. In this case, the cost of note insertion or deletion is clearly proportional to the note's duration. The replacement cost has two components, a cost proportional to the difference in durations (to make the durations equal), and a cost proportional to the pitch difference times minimum of the two

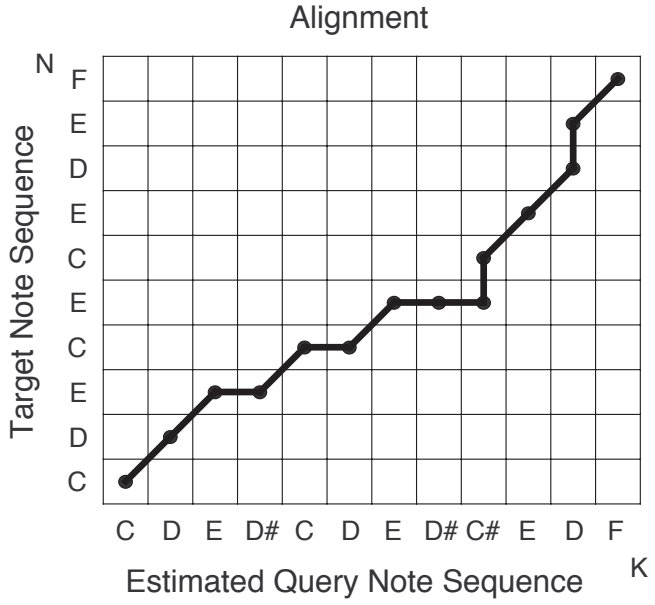

Figure 15: Sample string alignment for a query of "Do-Re-Me" and the true melody.

durations (to make the pitches equal, either before insertion or after deletion). We found $\alpha \approx 5$ to be a good value. Replacing two equal-duration notes with a pitch difference equal to α has equal cost to an insertion-deletion pair. Thus, our final costs are defined as

$$\begin{aligned} \text{Insert Cost}(\mathbf{t}_n) &= t_{n,\text{dur}} \\ \text{Delete Cost}(\mathbf{q}_k) &= q_{k,\text{dur}} \\ \text{Replace Cost}(\mathbf{q}_k, \mathbf{t}_n) &= |q_{k,\text{dur}} - t_{n,\text{dur}}| + \frac{2}{\alpha} \min(q_{k,\text{dur}}, t_{n,\text{dur}}) \, |q_{k,\text{pit}} - t_{n,\text{pit}}| \end{aligned} \tag{9}$$

We have explored other cost schemes as well, but no meaningful performance improvement was found. In particular, the cost scheme does not necessarily require separate insertion and deletion costs. By requiring every query note to be 'matched' to one target note, and vice versa, we arrive at the simpler cost scheme typically used by DTW algorithms, as described in section 3.3.1. Such a cost scheme was implemented, in which the match cost was simply proportional to the pitch difference. The performance of these two schemes is compared below.

4.3.2 Sequence Alignment

Fig. 15 shows an alignment path between a rather error-prone sample query of "Do-Re-Me" and the true theme. For simplicity, only note pitches are shown. In this figure, horizontal steps in the alignment path represent deleting an element from the query sequence, vertical steps represent inserting an element into the query sequence, and diagonal steps represent note replacement.

String matching is another dynamic alignment algorithm, and is identical to DTW. An optimal alignment between $\mathbf{Q}$ and $\mathbf{T}$ is found by recursively building a matrix $\mathbf{\Gamma} = [\gamma_{n,k}]$ of minimum prefix alignment costs. Let the alignment path be given by $\mathbf{w} = (w_1, w_2, \cdots w_T)$. The path must

be monotonic nondecreasing and adhere to the general local continuity constraint from Fig. 6; $\forall t,\ w_t - w_{t-1} \in \{(0,1),(1,1),(1,0)\}$. Starting in the lower left corner of $\mathbf{\Gamma}$, every element of $\mathbf{\Gamma}$ is found recursively by

$$\gamma_{n,k} = \min \begin{pmatrix} \gamma_{n-1,k} + \text{Insert Cost}(\mathbf{t}_n), \\ \gamma_{n,k-1} + \text{Delete Cost}(\mathbf{q}_k), \\ \gamma_{n-1,k-1} + \text{Replace Cost}(\mathbf{q}_k, \mathbf{t}_n) \end{pmatrix}. \tag{10}$$

The final alignment cost for $\mathbf{Q}$ and $\mathbf{T}$ is given by $\gamma_{N,K}$, and is interpreted as a dissimilarity measure. By performing this alignment on every target in the database we can rank order the target themes. The complexity of this alignment procedure is $O(NK)$.

To extract the actual alignment for warping the query to the target the algorithm must also construct a $N+1 \times K+1$ matrix $\mathbf{\Psi}$. The $(n,k)^{\text{th}}$ element of $\mathbf{\Psi}$ stores which of the three directions the lowest-cost path came from. Starting at $[\mathbf{\Psi}]_{N,K}$, the algorithm regresses back to $[\mathbf{\Psi}]_{0,0}$, tracing out the optimal alignment.

It is unreasonable to expect the query and correct target to be in the same key. To account for this, the mean pitch difference between the query and target note sequences (weighted by note lengths) is subtracted prior to alignment. In fact, the constant-shift and alignment is iterated to further improve the final match quality. While the initial constant-shift is essential, the iterative algorithm only improves performance modestly. This iteration is included in the results presented below.

4.4 Performance

Fig. 16 displays the classification accuracy and MRR of six QBH systems that employ a note representation across the same set of target databases as Fig. 7. The performance of five note estimators are given: a pitch-derivative segmenter, a RLS filter, a NLMS filter, a quantizer and an HMM. For all five note estimators, the cost scheme described in section 4.3.1 is used for sequence alignment. For the HMM estimator, a second cost scheme is included, the standard DTW cost scheme given given by (2). The curve representing the HMM note estimator with the DTW cost scheme is labelled "Note: HMM, DTW."

In [2, 3], numerous note estimators were compared using a single target database of fourteen themes; it was found that the baseline pitch-derivative yielded the worst performance by a considerable margin, the RLS and NLMS estimators yielded equivalent retrieval performance, and the quantizer and HMM yielded the best performance. The present work augments these results and notes some important differences that only become apparent as the target database size increases.

First, the RLS filter appears to yield essentially the equivalent performance as the baseline pitch-derivative. That is, the RLS filter yields only a 2% higher CA than the pitch derivative for all target database sizes. For the smallest target database size of fourteen 'authentic' themes, this difference appears meaningful. However, as the target database size increases, the performance of the RLS filter degrades as rapidly as the pitch-derivative, whereas the performance of the other note estimators degrades more slowly. In particular, the NLMS estimator, the one other estimator considered here that does not incorporate a 12-tone prior into segmentation, performs considerably better than the RLS and pitch-derivatives.

From Fig. 16, it is evident that incorporating the 12-tone constraint into note segmentation yields substantially more robust retrieval performance. Indeed, the quantizer and HMM yield $CA \approx 76\%$ for the largest target database, whereas the other note estimators yield $CA \approx 64\%$.

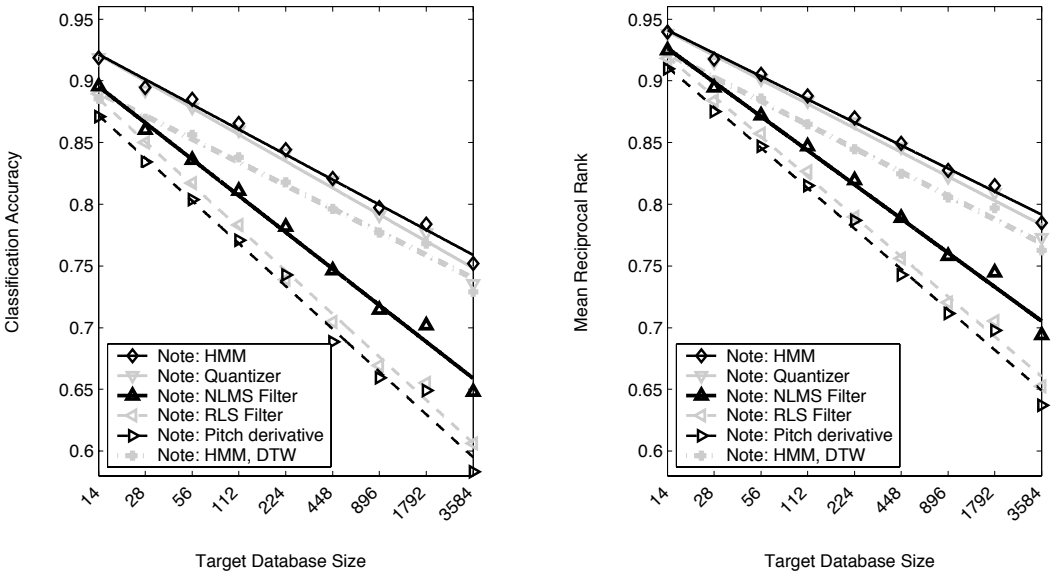

Figure 16: Performance for six QBH systems that employ a note representation vs. target database size. The left plot gives the classification accuracy and the right plot gives the MRR. All six QBH systems use a sequence of notes to represent the query and target for retrieval. For the curve labelled "Note: HMM, DTW", the cost scheme (4) is used, all others use (9).

This supplements the results presented in [2], in which it was concluded that an HMM yields the best retrieval performance. Given than the quantizer yields virtually the same performance as the HMM, it appears that incorporating the 12-tone constraint is the principle reason for the improved performance, not the structure of the HMM. Although, note that as the target database size increases, the HMM does yield marginally better performance than the quantizer.

Fig. 16 presents results for the HMM note estimator using both the cost scheme presented in section 4.3.1 and the standard DTW cost scheme of (2). As is evident from the figure, the DTW cost scheme performs somewhat worse than the cost scheme of section 4.3.1. We found that performance degrades a similar amount if the DTW cost scheme is applied to the other note estimators as well. It is curious however, that as the target database size increases, the performance of the DTW cost scheme degrades more slowly than the cost scheme presented in section 4.3.1. This would seem to imply that not explicitly allowing for note insertions and deletions[11] performs better as the target database is scaled to massive proportions. A similar comparison will be made in section 6 between the note and contour representations.

5 Histogram Representation

As will be shown in the next section, the contour representation yields more robust performance than the note representation. This comes at the cost of a much larger query representation, however.

[11]That is, requiring every query note to be *matched* to at least one target note, and vice versa.

Indeed, for our MATLAB implementation, aligning a pair of note sequences takes about 0.001s and aligning a pair of pitch contours takes about 0.1s. The contour representation is impractical for any real-world QBH system with a large target database. This observation motivates an alternative representation that yields similarly robust performance as the contour representation, but without the computational burden. We propose a novel sequence of pitch histograms.

Perhaps the most difficult component of sung query transcription is note segmentation [2, 3, 9, 28, 29]. For example, a portion of pitch contour that is slowly ascending can be labelled as either a slow transition between two notes or a sequence of rapid passing notes. There is an inherent tradeoff between insertion and deletion errors in any segmentation problem [3, 16]. Most QBH systems are implicitly tuned to have roughly equal note insertion and deletion rates. An interesting exception is presented in [34], in which a note estimator with a very high insertion rate is coupled with an alignment procedure that accounts for many insertion errors. As discussed in section 3, the pitch detection algorithm employed in the present work performs a partial segmentation, which can be interpreted as a note segmenter with a high note deletion rate[12]. In this case, each contour region represents one or more notes. Each region is collapsed into a single histogram of pitches. In so doing, we model the sung query as a partially ordered set[13].

Pitch histograms have been proposed before for MIR. Tzanetakis and Cooke [41] proposed pitch histograms for genre classification and Heo et. al. [13] and Song et. al. [39] have proposed sequences of pitch histograms for query-by-humming. There are some important distinctions however. In [13, 39] pitch histograms are employed to account for uncertainty in pitch detection as well as polyphonic sources. In contrast, we employ pitch histograms to discard ambiguous timing information within each contour region. Furthermore, [13,39] compute pitch histograms for a constant frame size, whereas we compute a single histogram for every contour region. In this case the duration represented by each histogram varies, and a modified DP alignment algorithm is required.

5.1 Query & Target Histograms

A sample query pitch-contour for the tune "Yankee Doodle" is shown in Figure 17, and the corresponding sequence of pitch histograms is shown in Figure 18. As is evident from the figures, this contour contains six regions and hence the histogram sequence is of length six.

Throughout this work a histogram bin width of $\delta = 0.2$ is used, hence there are 5 bins per half-step. Bin widths smaller than 0.2 do not improve performance substantially. Furthermore, $\delta > 0.2$ yields somewhat worse performance, although bin widths as large as 0.5 or 1 still yield reasonable performance.

Let the sequence of query histograms be given by $\mathbf{Q} = (\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_K)$, and similarly the target sequence be given by $\mathbf{T} = (\mathbf{t}_1, \mathbf{t}_2, \cdots, \mathbf{t}_N)$. Each query histogram is given by $\mathbf{q}_k = [q_k^1, q_k^2, \cdots, q_k^M]$, and every target histogram by $\mathbf{t}_n = [t_n^1, t_n^2, \cdots, t_n^M]$. The number of samples in the m^{th} bin of the k^{th} query histogram is given by q_k^m.

Before the query and target sequences are aligned, both sequences are normalized to unit 'vol-

[12]Indeed, applying such a note segmenter to our database of sung queries (that is, using the gaps in the pitch contour to indicate new notes) yields a segmentation with a deletion rate of about 35%. Note that this segmenter is similar to an amplitude threshold, short pauses are used to detect note boundaries.

[13]The order of contour regions is defined, but the order of pitches within each region is not.

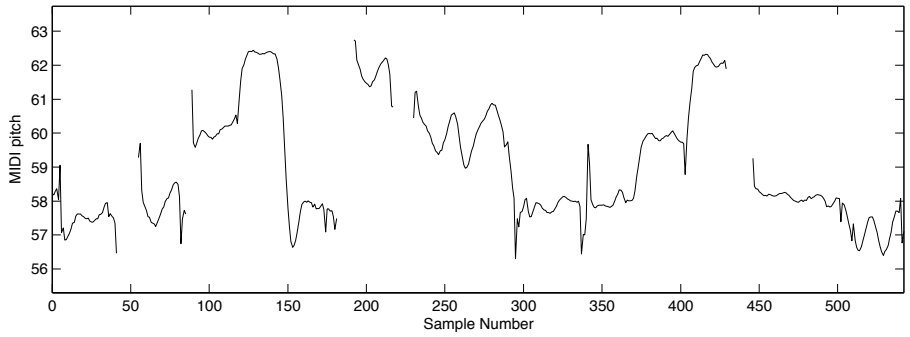

Figure 17: Sample query pitch-contour for “Yankee Doodle.”

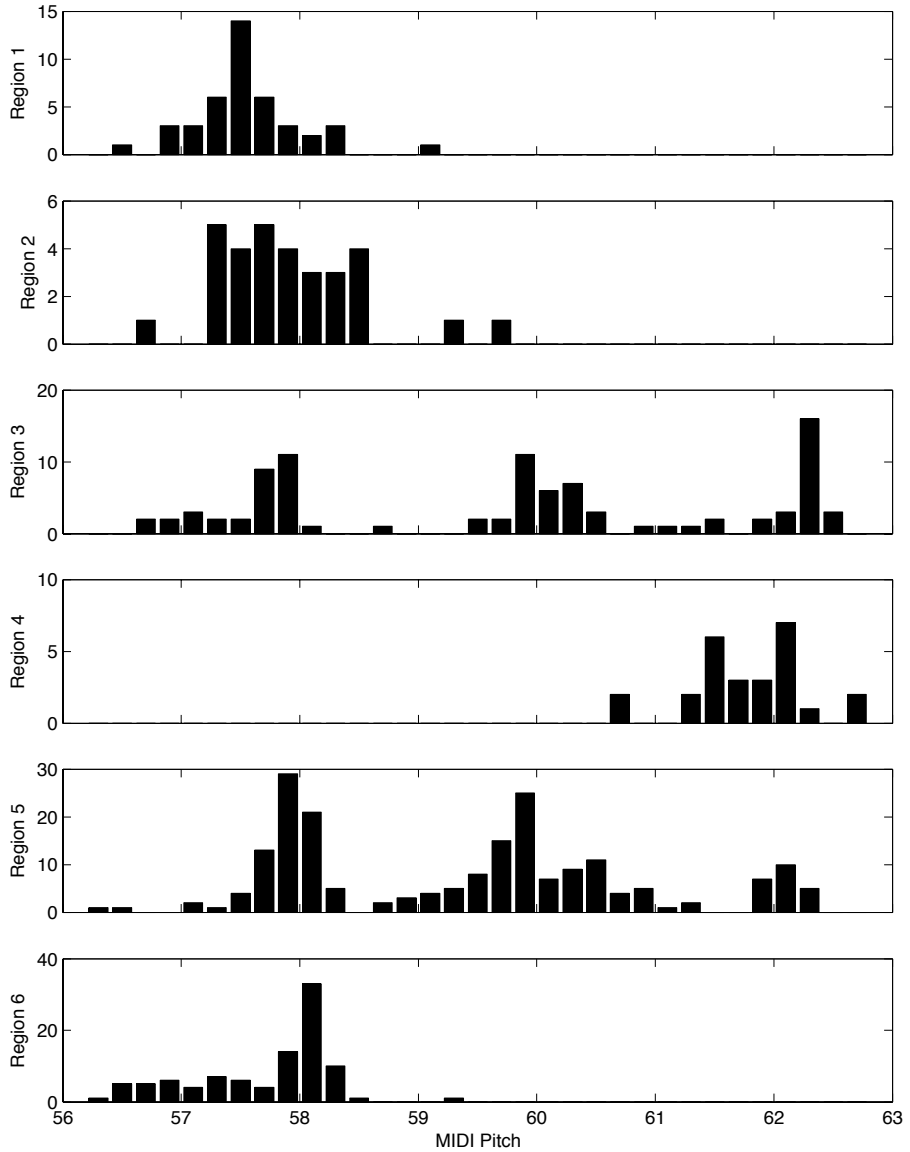

Figure 18: Pitch histograms for each region of contour shown in Figure 17.

ume,'

$$\sum_{k=1}^{K}\sum_{m=1}^{M} q_k^m = \sum_{n=1}^{N}\sum_{m=1}^{M} t_n^m = 1 \tag{11}$$

thus guaranteeing tempo-invariance. To make the system transposition-invariant, the mean pitch difference between the query and target *contours* is subtracted from the contours prior to computing the histograms, as is described in Section 3.3.2.

Note that the target sequence of histograms is represented using a separate histogram for every note; $\forall\ 1 \leq n \leq N,\ \exists!\ m$ S.T. $t_n^m > 0$. The length of the target sequence, N, equals the number of notes in the target. If a singer were to articulate each note individually with a 'da' then $K \approx N$. Because most users sing portions of the melody continuously, typically $K < N$. If the user sang completely legato, one continuous melody, then $K = 1$.

Unlike the systems presented in previous sections, it is not sufficient to only compare $\mathbf{q}_k$ and $\mathbf{t}_n$ when aligning $\mathbf{Q}$ and $\mathbf{T}$. Because each query histogram may represent more than one note, it is necessary to compare each query histogram to a collection of target histograms. Accordingly, let $\mathbf{t}_{(n,p)}$ be the cumulative sum of $\mathbf{t}_n$ through $\mathbf{t}_p$,

$$\begin{aligned} \mathbf{t}_{(n,p)} &= \sum_{i=n}^{p} \mathbf{t}_i \\ &= \left[\sum_{i=n}^{p} t_i^1,\ \sum_{i=n}^{p} t_i^2,\ \cdots, \sum_{i=n}^{p} t_i^M\right]. \end{aligned} \tag{12}$$

5.2 Match Cost

We use a musically intuitive match cost that shares some features with quantization error. That is, the match cost for $\mathbf{q}_k$ and $\mathbf{t}_{(n,p)}$ is computed by associating, or 'quantizing', every query bin to the nearest non-zero bin in $\mathbf{t}_{(n,p)}$. The query histogram is partitioned in cells based on $\mathbf{t}_{(n,p)}$. A cost is computed for each cell, and the final histogram match cost is given as the sum of the cell costs. A Voronoi partition is used to define cell boundaries[14]. The cost of each cell is given by two components, a duration difference and a quantization error. If the duration component is neglected, the cell cost would be similar to the quantization error of quantizing the query bins to the non-zero target bin.

Fig. 19 shows query histogram $\mathbf{q}_3$ for the pitch contour shown in Fig. 17, as well as the cumulative target histogram $\mathbf{t}_{(3,5)}$ for the main theme from "Yankee Doodle." The cell boundaries are shown with vertical lines. Note that all three target notes represented in this example have the same duration. Also note that the histograms in Fig. 19 have been normalized according to (11.

The cost of each cell is given by two components, a duration difference and a quantization error. If the duration component is neglected, the cell cost would be similar to the quantization error of quantizing the query bins to the non-zero target bin.

The cost for each cell is similar to the cost scheme outlined in Section 4.3.1, in that the are two components to the error: a duration component and a pitch component. If the duration component is neglected, the cell cost would be similar to the quantization error of quantizing the query bins to the non-zero target bin. Let m_0 be the non-zero bin in the current cell of the target histogram

[14]Cell boundaries are given by the midpoint between non-zero bins in $\mathbf{t}_{(n,p)}$. Hence the number of cells is equal to $p - n + 1$.

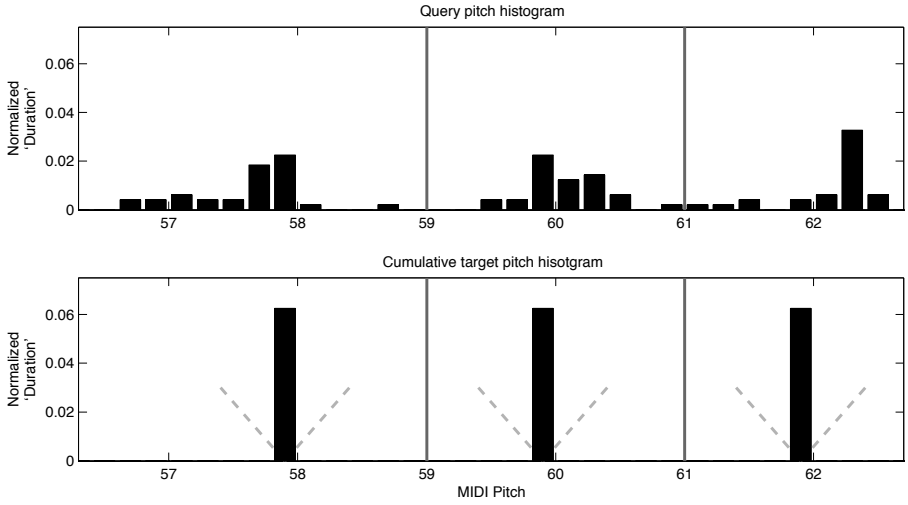

Figure 19: The top plot is the normalized query histogram $\mathbf{q}_3$ from Fig. 18. The bottom plot is the normalized cumulative target histogram $\mathbf{t}_{(3,5)}$ for the main theme from "Yankee Doodle." The vertical lines represent the cell boundaries for associating query bins with target bins and the dotted lines show the linear trend with which cost increases away from the non-zero target bins.

$\mathbf{t}_{(n,p)}$. The cost for the current cell is computed as follows. Beginning with bin m_0, 'quantize' query bins to the target bin. Radiate outward from m_0[15] until the total duration accrued by the 'quantized' query bins equals the duration of the target bin. For every query bin q_k^m 'quantized' to m_0, increment the cost according to

$$\text{Cell Cost} = \text{Cell Cost} + \alpha \cdot q_k^m \cdot |m - m_0|^p. \tag{13}$$

The $|m - m_0|$ radius term yields increasing error as query bins farther away from m_0 are 'quantized' with the target bin, and the q_k^m term accounts for the number of contour samples that are quantized to m_0. $p = 1$ was found to be a good value, as in Section 4.3.1. When either the accrued query duration equals the target duration, or all query bins in this cell have been 'quantized' to m_0, the remaining difference in duration is simply added to the Cell Cost. Note that every query bin contributes either to the pitch component or the duration component, but not both. $\alpha = \frac{\delta}{4}$ was found to be a good value.

5.3 Alignment Procedure

For the note and contour representations, every query element is matched to one of N possible target elements. For the histogram representation however, every query histogram, $\mathbf{q}_k$, is matched to one of N^2 possible cumulative target histograms, $\mathbf{t}_{(n,p)}$. To find the optimal alignment between sequences $\mathbf{Q}$ and $\mathbf{T}$ we construct a $N \times N \times K$ matrix $\mathbf{\Gamma} = [\gamma_{n,p,k}]$. The $(n, p, k)^{\text{th}}$ element of

[15]That is, begin with bin m_0, then 'quantize' bin $m_0 + 1$, then $m_0 - 1$, then $m_0 + 2$ $\cdots$, stopping if a cell boundary is reached.

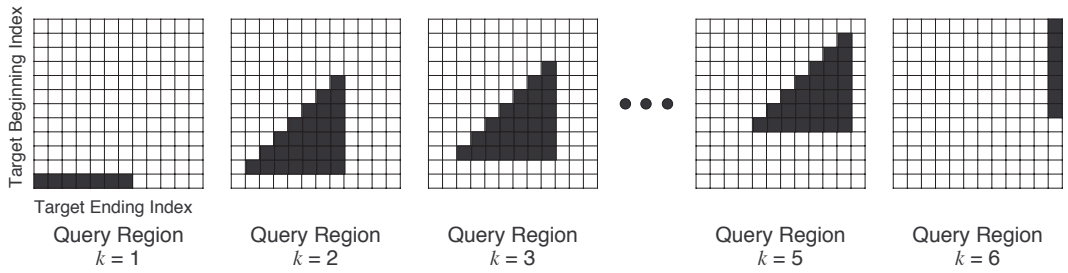

Figure 20: This figure shows the allowable alignment points in the coast matrix $\mathbf{\Gamma}$. Each panel shows one slice of the matrix, corresponding to one region of query pitch contour. Dark squares represent points the alignment path may visit. In this example the query is represented by a sequence of six histograms and the target by a sequence of twelve histograms.

$\mathbf{\Gamma}$ represents the minimum alignment cost for the prefix subsequences $(\mathbf{q}_1 \cdots \mathbf{q}_k)$ and $(\mathbf{t}_1 \cdots \mathbf{t}_p)$, with $\mathbf{q}_k$ matched to $\mathbf{t}_{(n,p)}$. Starting with $k = n = 1$, every element of $\mathbf{\Gamma}$ is found recursively by

$$\gamma_{n,p,k} = \min_r \gamma_{r,n-1,k-1} + \text{Match Cost}(\mathbf{q}_k, \mathbf{t}_{(n,p)}). \tag{14}$$

Figure 20 shows an example of the points in $\mathbf{\Gamma}$ that the alignment path can visit. This figure shows the case for $N = 12$ and $K = 6$. The dark squares in the k^{th} panel of the figure show all the possible cumulative target histograms that the k^{th} query histogram can be matched to. For each panel, the vertical index gives the starting point n for the cumulative histogram and the horizontal index gives the ending point p. For example, the first query histogram, $k = 1$, must match to a cumulative target histogram beginning with $n = 1$, but can end anywhere from $p = 1$ to $p = 7$ ($p = 8 \cdots 12$ is disallowed because at least five target histograms are needed to match to the remaining five query histograms).

Note the critical assumption, that every histogram aligns with at least one target histogram. This alignment algorithm is only valid for $N \geq K$. If for a particular query and target $N < K$, $K - N$ all-zero histograms are appended to $\mathbf{T}$ for the alignment algorithm to find an alignment path[16]. Unlike the alignment algorithms presented in previous sections, it is unclear how to define the 'slope' of the alignment path for this representation, hence no cost scheme that penalize extreme slopes are considered. It is nonetheless possible that more subtle cost schemes can be developed to improve retrieval performance.

5.4 Implementation Issues

The histogram representation is implemented both with and without extending query contour regions. As is evident from the Fig. 4, the observed query pitch contour often contains gaps in the contour, whereas the target pitch contour is defined for every time step. The target contour is time-scaled to be the same duration as the query contour prior to computing $\mathbf{Q}$ and $\mathbf{T}$. Hence more contour values are included in the target sequence of histograms than the query sequence. To account for this, the 'null' regions in the query contour are filled in, as described in section 3.3.2.

[16]While this solution is somewhat dubious, any target theme for which $N < K$ is not likely to be the true theme sung by the user.

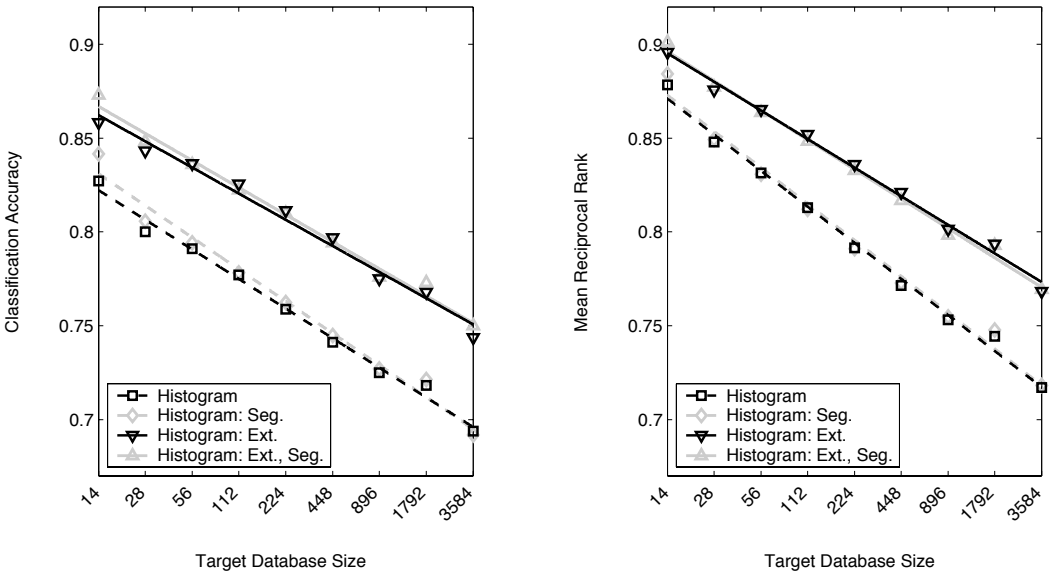

Figure 21: Performance of the four QBH systems that emply the histogram representation vs. target database size. The left plot gives the classification accuracy and the right plot gives the MRR.

The crude segmentation performed by the pitch detection algorithm can be augmented by any of the note segmenters described in the previous section, thus reducing the number of notes that are collapsed into a single histogram. All of the methods presented in Section 4 can be tuned to tradeoff inserted notes for deleted notes. For the histogram representation, the crude segmentation is refined somewhat by the NLMS segmenter tuned so that the note insertion rate is essentially zero whereas the deletion rate is relatively high[17]. While such a segmenter is ill-suited for a note representation, the histogram representation assumes that each query histogram represents at least one note.

5.5 Performance

Fig. 21 displays the classification accuracy and MRR of four QBH systems that employ the histogram representation across the same set of target databases as Fig. 7. The curves labelled "Ext." extend the query contour regions prior to computing the histogram sequence, as described above. The curves labelled "Seg." augment the segmentation performed by the pitch detection algorithm with a NLMS segmenter, as described above.

Clearly, filling in the 'null' query contour regions prior to computing the histogram sequence improves performance considerably. This is not surprising, the match cost described in section 5.2 assumes that equal number of contour samples are represented by the query and target histogram sequences.

As can be seen in the Fig. 21, the additional segmentation does not change performance. One

[17]In this case the note deletion rate is about 25%, rather than 35% if only the segmentation provided by the pitch detection algorithm is used

possible explanation is that the assumption that every query histogram represents at least one complete note is, in fact, dubious. While this assumption may be true for the vast majority of recorded queries in our database, about 10% of the test queries contain either spurious 'bursts' of pitch contour or occasionally split the contour within a single note. Both of these phenomena manifest themselves in 'poor' queries. Some queries contain background noise that the pitch detection algorithm occasionally detects a pitch in, especially at the beginning or end of a query. Many of these spurious bursts can be removed by simply discarding any region less than 70ms in length, but some remain. Furthermore, untrained singers will sometimes articulate pitch very poorly and the pitch detection algorithm will not detect a pitch at every frame throughout the duration of a note. Both of these problems could be mitigated by judicious pre-processing of the observed contour, improving performance for systems both with and without the extra segmentation.

6 Discussion

Fig. 22 gives a summary of the results presented in sections 3, 4, & 5. For both plots, target database size is represented along the abscissa. The ordinate represents classification accuracy for the left plot and MRR for the right plot. In both plots, three curves are included for the contour representation: direct comparison (without alignment), and the Itakura and Sakoe constraints (the two best alignment constraints). Three curves are included for the note representation: the pitch-derivative estimator (the poorest note estimator), and the quantizer and HMM estimators (the two best note estimators). One curve is included for the histogram representation. The contour representation curves are represented with a solid line, the note representation curves are represented with a dashed line, and the histogram representation curve is represented with a dash-dot line.

The relative performance of the various QBH methods demonstrate several interesting trends. As expected, direct comparison of pitch contours, without alignment, yields the poorest performance. It is striking however that using a common note estimator *with* alignment only yields marginally better performance, and that this improvement quickly vanishes as target database size increases. That is, melody transcription coupled with alignment does not necessarily perform any better than Euclidean distance applied directly to the pitch contours.

The quantizer and HMM note estimators yield considerably better performance than the pitch derivative estimator. Indeed, for a small target database size, the note representation computed using the quantizer and HMM estimators yield the best performance, $CA \approx 92\%$. However, the rate of performance degradation of the note representations is considerably faster than that of the best contour representations. The contour representation using the Itakura and Sakoe continuity constraints yield the most robust performance, the Sakoe constraint in particular. For the largest target database size the best contour representations yield $CA \approx 80\%$ whereas for the best note representations $CA \approx 75\%$. That the contour representation is more robust to increasing target database size is not surprising; as the number of targets grows, targets placed in ~ 10-dimensional space will inevitably be closer than targets in ~ 100-dimensional space.

The histogram representation does not outperform the best note or contour representations. However, the rate of performance degradation for the histogram representation is considerably slower than that of the note representations. The CA slope for the histogram representation is equal to that of the contour representation using the Itakura continuity constraint. For the largest target database, the histogram representation yields equal CA as the best note representation. This is an intriguing observation because while the contour representation yields the most robust per-

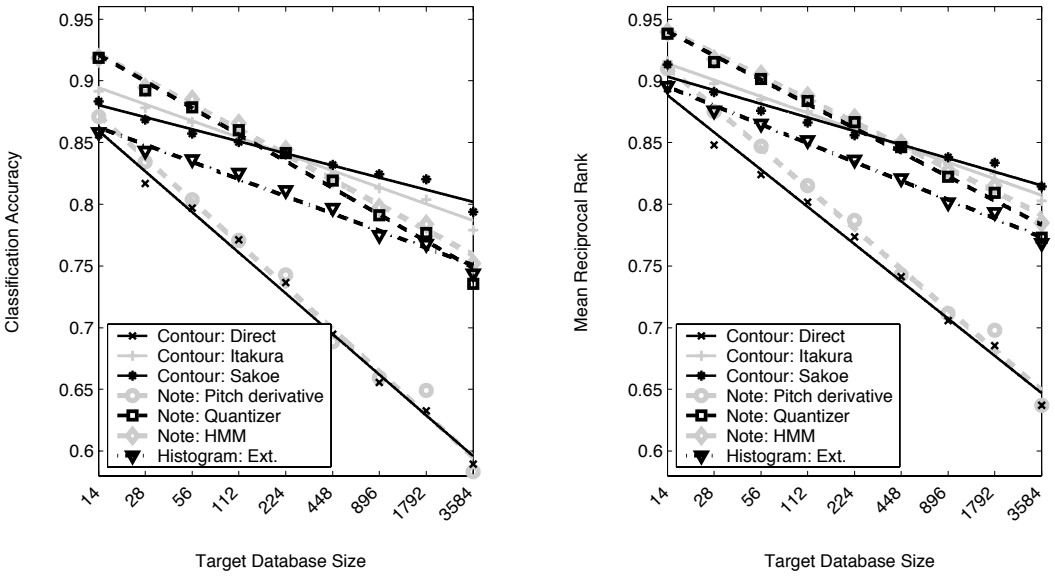

Figure 22: Performance of various QBH methods vs. target database size. Systems that employ a contour representation are shown with a solid line. Systems that employ a note representation are shown with a dashed line. The histogram representation is shown with a dash-dot line.

formance, it is computationally burdensome. For our MATLAB implementation, alignment of histogram sequences is only modestly more time consuming than for the note representation[18]. This implies that as the target database is scaled to massive proportions, the histogram representation could provide the best compromise between performance and retrieval speed.

The performance of the histogram representation can possibly be improved by more judicious 'continuity' constraints and histogram match costs. For the contour representation, the best performance is achieved with the Itakura and Sakoe continuity constraints. No alternative continuity constraints were explored for the histogram representation, there being no obvious interpretation for the 'slope' of the alignment path. Furthermore, the alignment procedure for the histogram representation assumes that every query histogram represents at least one target note, insertion errors are disallowed. Some of the query pitch contours contain spurious contour regions which result from environmental noise, however. Many of these spurious regions could be discarded using a minimum duration constraint, but those that remain cause the alignment procedure to find implausible alignments.

Comparing the CA and MRR results in Fig. 22, the note representations yield relatively better performance in terms of MRR than CA. Indeed, for the largest target database, the median rank of all misclassified queries using the note representation is about 30, whereas the median rank is about 100 for the contour representation. This is due to the different cost schemes. The contour and histogram cost schemes do not explicitly allow for portions of the query to simply be 'deleted'

[18]For our MATLAB implementation, aligning a pair of note sequences takes about 0.003s. Indeed, for the histogram representation, $\mathbf{\Gamma}$ contains $\sim 10^3$ elements. Whereas for the note and contour representations, $\mathbf{\Gamma}$ contains $\sim 10^2$ and $\sim 10^4$ elements, respectively.

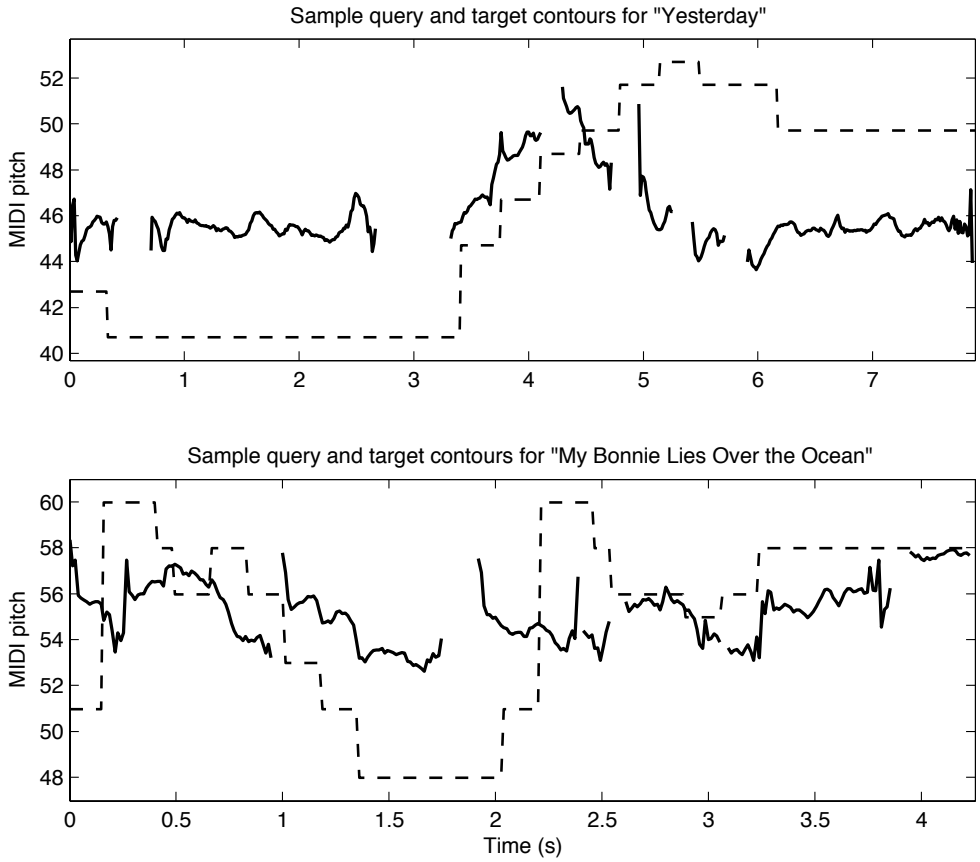

Figure 23: Sample query and target pitch contours for two particularly poor queries.

or 'inserted,' every element of the query must be matched to at least one element in the target. This occasionally results in pathological alignments for the contour and histogram representations, radically warping the query sequence to account for a short portion of incongruous data. In such situations the note representation alignment simply deletes or inserts the appropriate element.

6.1 Query Quality

As described in section 2.1, we employ a query test set of 480 naturally sung melodies. Figures 4, 9, & 17 give examples of 'good' query pitch contours. Not all of the queries in our database have such easily identifiable contours however. Figure 23 gives two examples of 'poor' query contours, along with the correct target contours. These two queries are virtually monotone, rendering a 'flattened' contour[19].

The contours shown in Figure 23 are exceptionally poor. There is, of course, a continuum

[19]In fact, we observed for our query test set that when a participant sang a note with incorrect pitch, the pitch they did sing was almost always a pitch they had sung earlier in the same query. This would seem to imply that an untrained singer would rather return to a pitch they 'knew' how to sing rather than waffle around a new pitch they are uncertain of (even if the pitch is wrong, at least it fits into a equal-tempered scale), implying a model for singer error.

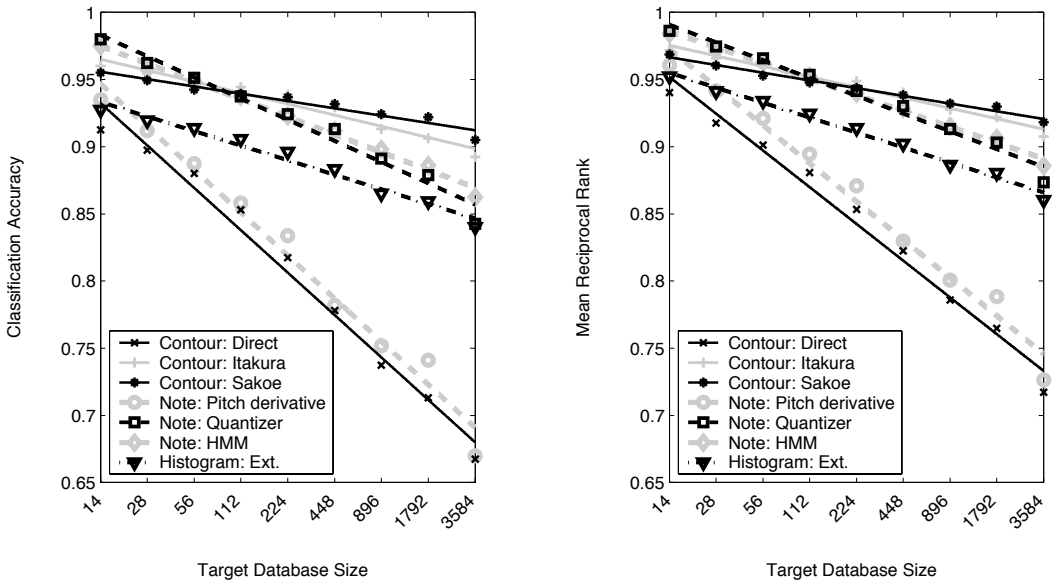

Figure 24: Performance of various QBH methods vs. target database size when excluding the poor queries in the test set.

in query contour quality between these poor examples and the relatively good examples shown earlier. It is unclear that any QBH system should be expected to perform well even with such bad queries. Specific modeling of singer and transcription error is becoming a more active area of QBH research [29, 34].

It should be noted that the author has no difficulty classifying either query shown in Fig. 23, albeit this is due to the author's knowledge of the lyrics of both themes. As mentioned at the outset, our query representations are based exclusively on the estimated pitch contour. Employing other features such as broad spectral or phonetic information may further improve performance.

We therefore consider the performance of the QBH systems on a reduced database, discarding the 'poor' queries. Figure 24 gives the performance for the QBH systems shown in Figure 22, but with all queries from three singers discarded. These three of the fifteen participants sang virtually monotone, and often very fast. Discarding the queries from these three participants left a test set of 400 queries.

There are two noteworthy differences in performance between Fig. 24 and Fig. 22. First, the contour representation performs relatively well on the filtered test set compared to the note representation. This reinforces the earlier observation that the contour representation does not perform well on queries that contain singer errors. Second, the histogram representation performs relatively poorly on the filtered test set compared to the contour and note representations. This implies that the histogram representation is relatively adept at classifying poor quality queries.

7 Conclusion

This work gave a thorough comparison of three query representations in the context of a QBH system. Two representations have been previously proposed: a sequence of notes and a pitch contour. A novel query representation is also proposed, a sequence of pitch histograms. Five note estimators were considered for the note representation, and it was found that estimators that incorporate a 12-tone prior into note segmentation yield the most robust performance. Several implementations were considered for the contour representation, and it was found that continuity constraints and cost schemes that penalize extreme alignment slopes yield the most robust performance. While the contour representation was found to yield more robust performance than the note representation, this comes at the cost of greatly increased computational complexity. A piecewise linear approximation for the contour representation was proposed to reduce the size of the representation, but the implementation was found to perform poorly.

A novel sequence of histograms was proposed. This representation collapses separate regions of pitch contour into a single histogram. This representation required a modified dynamic alignment procedure for retrieval. This representation was found to be modestly slower than the note representation, but provided, in some sense, more robust performance. The histogram representation presents an potentially useful compromise between note and contour representations.

References

[1] N.H. Adams, *Automatic Segmentation of Sung Melodies*, Technical Report, University of Michigan (2002), http://www.eecs.umich.edu/techreports/systems/cspl/cspl-340.ps.gz.

[2] N.H. Adams, M. Bartch, and G.H. Wakefield, *Coding of Sung Queries for Music Information Retreival*, IEEE Workshop on Application of Signal Processing to Audio and Acoustics (2003), New Paltz, NY.

[3] N.H. Adams, M.A. Bartch, and G.H. Wakefield, *Note Segmentation and Quantization for Music Information Retreival*, *submitted to* IEEE Transactions on Speech and Audio Processing.

[4] M.A. Bartsch, *Automatic Assessment of the Spasmodic Voice*, Technical Report, University of Michigan (2002), http://www.eecs.umich.edu/techreports/systems/cspl/cspl-339.ps.gz.

[5] W. P. Birmingham, R. B. Dannenberg, G. H. Wakefield, M. Bartsch D. Bykowski, D. Mazzoni, C. Meek, M. Mellody, and W. Rand, *Musart: Music Retrieval Via Aural Queries*, Proceedings of ISMIR 2001 (2001), Bloomington, IN.

[6] Paul Boersma, *Accurate Short-Term Analysis of the Fundamental Frequency and the Harmonics-to-Noise Ratio of a Sampled Sound*, Proceedings of the Institute of Phonetic Sciences of the University of Amsterdam, vol. 17, 1993, pp. 97–110.

[7] P. Cano, E. Batlle, H. Mayer, and H. Neuschmied, *Robust Sound Modeling for Song Identification in Broadcast Audio*, AES Convention (Munich, Germany), May 2002.

[8] 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).

[9] R. B. Dannenberg, W. P. Birmingham, and G. Tzanetakis et. al., *The MUSART Testbed for Query-By-Humming Evaluation*, Proceedings of ISMIR, 2003, Baltimore, MD.

[10] J. S. Downie, *Toward the Scientific Evaluation of Music Information Retrieval Systems*, Proceedings of ISMIR, 2003, Baltimore, MD.

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

[12] D. Gusfield, *Algorithms on Strings, Trees and Sequences*, Cambridge University Press, Cambridge, UK, 1999.

[13] S. P. Heo, M. Suzuki, A. Ito, and S. Makino, *Three Dimensional Continuous DP Algorithm for Multiple Pitch Candidates in Music Information Retrieval System*, Proceedings of ISMIR, 2003, Baltimore, MD.

[14] F. Itakura, *Minimum Prediction Residual Principle Applied to Speech Recognition*, IEEE Transactions on Acoustics, Speech and Signal Processing **23** (1975), no. 1, 67–72.

[15] T. Kageyama, K. Mochizuki, and Y. Takashima, *Melody Retrieval with Humming*, Proceedings of Int. Computer Music Conference (ICMC), 1993, Tokyo, Japan.

[16] S. Kay, *Fundamentals of Statistical Signal Processing: Detection Theory*, Prentice Hall Ptr, Upper Saddle River, NJ, 1998.

[17] E. Keogh, *Exact Indexing of Dynamic Time Warping*, Proceedings of the 28th VLDB Conference (2002).

[18] 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), 263–286.

[19] E. Keogh, S. Chu, D. Hart, and M. Pazzani, *Segmenting Time Series: A Survey and Novel Approach*, Data Mining in Time Series Databases, World Scientific Publishing Company, 1993.

[20] E. Keogh and M. Pazzani, *Scaling up Dynamic Time Warping to Massive Datasets*, Proceedings of the 3rd European Conference on Principles and Practice of Knowledge Discovery in Databases (1999), 1–11.

[21] E. Keogh and P. Smyth, *A Probabilistic Approach to Fast Pattern Matching in Time Series Databases*, Proceedings of the 3rd International Conference of Knowledge Discovery and Data Mining (1997), 24–29.

[22] N. Kosugi, Y. Nishihara, and T. Sakata et al, *A PracticalL Query-By-Humming System for a Large Music Database*, ACM Multimedia (2000), 333–342, Los Angelos, CA.

[23] B. Liu, Y. Wu, and Y. Li, *A Linear Hidden Markov Model for Music Information Retrieval Based on Humming*, Proc. ICASSP'03, 2003.

[24] D. Mazzoni, *Tech Report: Melody Matching Using Time Warping*, Technical Report, Carnegie Mellon University (2002).

[25] D. Mazzoni and R. B. Dannenberg, *Melody Matching Directly from Audio*, Proceedings of ISMIR 2001 (2001), Bloomington, IN.

[26] R. J. McNab and L. A. Smith, *Evaluation of a Melody Transcription System*, IEEE International Conference on Multimedia and Expo, 2000 **2** (2000), 819–822.

[27] R. J. McNab, L. A. Smith, D. Bainbridge, and I. H. Witten, *The New Zealand Digital Library MELody inDEX*, D-Lib Magazine (1997), http://www.dlib.org/dlib/may97/meldex/05witten.html.

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

[29] C. Meek and W. Birmingham, *Johnny can't sing: A comprehensive error model for sung music queries*, Proceedings of ISMIR (2002).

[30] M. Mellody, M.A. Bartsch, and G.H. Wakefield, *Analysis of Vowels in Sung Queries for a Music Information Retrieval System*, Journal of Intelligent Information Systems **21** (2003), no. 1, 35–52.

[31] D. Parsons, *The Directory of Tunes*, Spencer Brown and Co., Cambridge, England, 1975.

[32] S. Pauws, *Cubyhum: A Fully Operational Query by Humming System*, Proceedings of ISMIR, 2002, Paris, France, pp. 187–196.

[33] T. Pavlidis, *Waveform Segmentation Through Functional Approximation*, IEEE Transactions on Computers **C-22** (1973), no. 7, 689–697.

[34] A. Pikrakis, S. Theodoridis, and D. Kamarotos, *Recognition of Isolated Musical Patterns Using Context Dependent Dynamic Time Warping*, IEEE Transactions on Speech and Audio Processing **11** (2003), no. 3, 175–183.

[35] L.R. Rabiner and B.H. Juang, *An Introduction to Hidden Markov Models*, IEEE ASSP Magazine (1986), 5–16.

[36] ———, *Fundamentals of Speech Recognition*, Prentice Hall, Upper Saddle River, NJ, 1993.

[37] H. Sakoe and S. Chiba, *Dynamic Programming Algorithm Optimization for Spoken Word Recognition*, IEEE Transactions on Acoustics, Speech and Signal Processing **26** (1978), no. 1, 43–49.

[38] J. Shifrin, B. Pardo, C. Meek, and W.P. Birmingham, *HMM-based musical query retrieval*, International Conference on Digital Libraries (Portland, OR), 2002, pp. 295–300.

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

[40] T. Tolonen and M. Karjalainen, *A Computationally Efficient Multi-Pitch Analysis Model*, IEEE Transactions on Speech and Audio Processing **8** (2000), no. 6.

[41] G. Tzanetakis, A. Ermolinskyi, and P. Cook, *Pitch Histograms in Audio and Symbolic Music Information Retrieval*, Proceedings of ISMIR, 2002, Paris, France.

[42] R. Yaniv and D. Burshtein, *An Enhanced Dynamic Time Warping Model for Improved Estimation of DTW Parameters*, IEEE Transactions on Speech and Audio Processing **11** (2003), no. 3, 216–228.

[43] B.K. Yi, H. Jagadish, and C. Faloustsos, *Efficient Retrieval of Similar Time Sequences Under Time Warping*, Proc. IEEE International Conference on Data Engineering, 1998, pp. 201–208.