Journal of Intelligent Information Systems, 21:1, 35–52, 2003

Analysis of Vowels in Sung Queries for a Music Information Retrieval System

MAUREEN MELLODY maureen@stievater.com
Applied Physics Program, University of Michigan, Ann Arbor, Michigan, USA

MARK A. BARTSCH* mbartsch@eecs.umich.edu
GREGORY H. WAKEFIELD ghw@eecs.umich.edu
Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, Michigan, USA

Received June 30, 2002; Revised July 31, 2002; Accepted August 15, 2002

Abstract. A method for analyzing and categorizing the vowels of a sung query is described and analyzed. This query system uses a combination of spectral analysis and parametric clustering techniques to divide a single query into different vowel regions. The method is applied separately to each query, so no training or repeated measures are necessary. The vowel regions are then transformed into strings and string search methods are used to compare the results from various songs. We apply this method to a small pilot study consisting of 40 sung queries from each of 7 songs. Approximately 60% of the queries are correctly identified with their corresponding song, using only the vowel stream as the identifier.

Keywords: music information retrieval, vowels, singing, query by humming

1. Introduction

One can easily envision the following scenario: a music listener has heard a pop tune on the radio, and would like to download a copy from a computer database of musical pieces. However, he knows neither the name of the song nor its singer, but can remember some of the music itself. The most convenient method for this listener to query the music database is to sing the remembered portion of the pop tune into his computer. The quickly growing field of music information retrieval often uses such a "query-by-humming" paradigm for searching musical databases. In most such systems, a fundamental frequency tracking algorithm is used to parse a sung query for melodic content (McNab et al., 1996; Smith et al., 1997; Mazzoni and Dannenberg, 2001). The resulting melodic information is used to search a musical database using either string matching techniques (Ghias et al., 1995; Smith et al., 1997; McNab et al., 2000) or other models such as hidden Markov models (Birmingham et al., 2001; Shifrin et al., 2002).

What happens, though, if the person singing into the retrieval system is a terrible singer? One approach that is being investigated is the development of sophisticated models of pitch error (Meek and Birmingham, 2002). Such models, however, still assume that the user of the system has some moderate singing ability. This is a general flaw with the use of singing

*Author to whom all correspondence should be addressed.

(or humming, or whistling) as the primary method of music information retrieval: what happens if the user cannot "carry a tune"? In these instances, a system may be able to increase performance by analyze and evaluate sung lyrics along with pitch and rhythm.

Specifically, we would like to identify the vowels that the user sings, since the vowels comprise the majority of time in singing (Miller, 1996). These vowels can then be used in conjunction with pitch and rhythm as separate streams of information in the user's query. The most robust music query system is one that is user-independent. Therefore, it is desirable to avoid training a query system to respond to a particular voice. For this reason, conventional speech recognition algorithms are not applicable, as typically these systems require training a hidden Markov model with 40 or more observations (Rabiner, 1989). Additionally, these methods of vowel identification are likely to fail when the pitch of the query starts to vary by an octave or more, as is common in singing but extremely uncommon in speech. For these reasons, we explore a *self-similar* method of vowel identification, in which vowel content is compared within a single query for the purposes of identification. In this self-similar evaluation, the sung vowels are compared to each other within the ongoing vocalic stream, rather than being compared to a specific template. This has the advantage in that different styles of music and accent affect the vowel coloration, e.g., country-western vs. Italian opera. In addition, as the self-comparison is within a single sung phrase, there is no need for training.

In this paper we develop the structure of self-similar vowel analysis and illustrate its application to a set of sung queries. Results are presented from a study involving a small number of singers producing a database of a few common, well-known songs. These results are discussed with respect to scaling the method up to a production-level song-query database.

2. Vowel analysis method

In the following, we describe the method of self-similar vowel analysis in a tutorial format, through its use in a small study. The general framework for this method of vowel analysis can then be applied to a larger data set.

2.1. Query recordings

A total of 7 songs were sung 4 times each by every user. Songs common to the United States were chosen: "Happy Birthday," "Yankee Doodle," "America the Beautiful," the Beatles' "Yesterday," "Row, Row, Row Your Boat," "Somewhere Over the Rainbow," and "My Bonnie Lies Over the Ocean." Only the first phrase of each song was used in an attempt to keep the input queries at roughly the same length. The lyrics for each song were provided to the singer, but no pitch information was given. The users were given no other instructions as to musical performance.

Users were recorded in a quiet environment and were alone throughout the recording session. Each recording required roughly 20 minutes. A Sony MZ-R55 portable minidisc recorder and an Audio-Technica AT831b lavaliere microphone were used to record the sung passages. The microphone was attached to the user at the level of their sternum. Initial sampling rate was 44.1 kHz. The data were transferred digitally to computer where they were downsampled to 8 kHz for analysis.

2.2. *User population*

Users were volunteers who were drawn primarily from the graduate student population in the EECS department at the University of Michigan. A total of 10 singers were recorded, with 7 male and 3 female, ranging in age from 23 to 41. The singers were not screened in any way. Six of the users had some musical background, while 4 had no musical experience. All were native English speakers.

2.3. *Signal analysis methods*

A spectrogram is used to obtained information about the distribution of signal energy over time and frequency. Each time splice in the spectrogram is normalized to have the same average power. For each time slice, the fundamental frequency of the sustained vowel f_0 is estimated. This allows the amplitude of the harmonic partials to be located with a simple peak-picking algorithm which searches for local maxima on the spectrogram. Both the size and location of the regions searched for harmonic partials is dependent upon the fundamental frequency. Specifically, given a time slice of the spectrogram $S(f)$ in decibels, the amplitude of the kth harmonic partial is calculated as

$$A_k = \max\{S(f)\}, \quad f_0(k - 1/2) \leq f < f_0(k + 1/2). \tag{1}$$

The amplitude of the harmonic partials within each frame are converted to a vector with fixed dimension by linearly interpolating between neighboring partial amplitudes (in dB) across frequency.

In the present study, we performed a spectrogram analysis on each 8 kHz recording, using a 512-point Hamming window and a 512-point FFT with an overlap of 307 points. This corresponds to a data window of 64 ms at intervals of 26 ms. Fundamental frequency used for the pitch-adaptive method of partial extraction was determined by using a standard spectral contraction method for pitch detection (Hermes, 1993). Interpolation over the peaks was performed over a 512-point evenly spaced grid of frequencies.

2.4. *Data clustering methods*

Once the signal has been broken into successive slices of interpolated spectra, clustering is performed to segment the stream into vowel groups using the nearest mean reclassification algorithm (NMRA) (Fukunaga, 1990). In this algorithm, the number of clusters is specified in advance; otherwise, clustering is performed without any strong assumptions concerning the structure of the data. In brief, NMRA proceeds as follows:

1. Choose the order of the cluster model, L.
2. Randomly assign each data vector $\mathbf{v_i}$ to a cluster. Let the label of the ith vector be $C(i)$, so $C(i) \in \{1 \dots L\}$.

3. Find the mean vector for the jth cluster, $\bar{\mathbf{v}}_j$, as

$$\bar{\mathbf{v}}_j = \frac{1}{|u|} \sum_{k \in u} \mathbf{v}_k, \qquad u = \{i : C(i) = j\} \tag{2}$$

   where $|u|$ is the cardinality of the set u (i.e., number of elements in u).
4. Re-classify each data vector to the cluster with the nearest mean vector, using a Euclidean norm as

$$C(i) = \underset{j}{\operatorname{argmin}} (\bar{\mathbf{v}}_j - \mathbf{v}_i)^T (\bar{\mathbf{v}}_j - \mathbf{v}_i) \tag{3}$$

5. Repeat steps 3–4 until the new clustering solution is the same as the previous clustering solution.

In the present study, the number of clusters, L, is chosen to be four. This model order was chosen empirically to generate enough clusters for basic vowel separation without too many to cause problems with idiosyncratic vowel production. In general, the four clusters tend to be: no vowel (either silence or a consonant), front (/i/-like) vowels, back (/u/-like) vowels, and neutral (/e/-like) vowels. This coarse vowel representation (4 instead of the typical 15 or so accepted vowels in Western singing (Titze, 1994)) provides more flexibility across the range of voices and song production in identifying the sung vowel stream.

After every NMRA evaluation of a query, the clusters are sorted so that the NMRA algorithm gives a consistent label to each of the identified vowel states. The clusters are sorted by the second spectral moment of the mean cluster vector. For a vector $\mathbf{v}$, we compute the second spectral moment as

$$m = \frac{1}{K} \sum_{k=1}^{K} \left(v_k - \frac{1}{K} \sum_{j=1}^{K} v_j \right)^2 \tag{4}$$

where K is the dimension of $\mathbf{v}$ and v_i indicates the ith element of of the vector $\mathbf{v}$. Since the spectral content of noise/silence is relatively flat, the second spectral moment is the smallest. As the vowel /i/ has a strong high-frequency second formant (2000–4000 Hz), its second spectral moment is also somewhat low. The /u/ vowel, whose first and second formants tend to be in the low frequency range ($<$1000 Hz), has the largest second spectral moment.

The NMRA algorithm relies upon a random initial configuration for clustering. Therefore, the final result is dependent upon this random initial configuration. To avoid obtaining an unlikely solution based upon a poor choice of initial configuration, the NMRA algorithm is run a total of 30 times for each data set. The most frequently occurring clustering solution is chosen as the final result for that data set. The number of runs was determined empirically; repeated runs of the NMRA algorithm above 30 did not seem to alter the mode of the solution.

This method of self-similar evaluation of vowels does not work on all types of input queries. For instance, one can easily imagine this method failing if the user sang only one vowel as the query. In this case, the NMRA algorithm would still seek to divide this query into 4 distinct classes. (The NMRA algorithm does not require that each cluster contain data. However, in all cases evaluated here, all clusters did have at least one member).

To avoid analyzing a query that does not fall within the self-similar, four-cluster model, a series of warning tests are evaluated using the NMRA results. These warning thresholds were all developed empirically from test data generated to resemble a genuine query. The warnings are as follows:

1. The overall duration of the input signal must contain at least 100 time slices. In other words, under the current settings, a query must be at least 2.6 seconds long.
2. Each cluster in the NMRA solution must contain at least 15 time slices.
3. No more than 20% of the NMRA solutions can differ from the mode solution. In other words, under the current settings, only 6 out of the 30 NMRA solutions for one query can be different from the final chosen solution.

2.5. String matching

In order to compare queries to songs that exist in a database, we have chosen to represent our songs as strings. Two strings are then compared using an edit distance metric, which determines the number of insertions, deletions, and replacements necessary to convert one string into another (Needleman and Wunsch, 1970). Because we cannot know how much of a song will be included in a query, we employ a modified sub-string-based edit distance calculation. Under this modification, we do not penalize deletions from the beginning or end of the database string. Thus, if a query contains only a few phrases from the middle of a song, the resulting query is not penalized because of its length or position in the song.

Rather than operating directly on the results of vowel string clustering, we use these results to produce a simplified string that can account for differing rates of production. The NMRA algorithm returns a stream of numbers to represent a given query, where each time slice is given a number that corresponds to a vowel cluster label. Instead of using this stream directly, we form a reduced eight-symbol representation: one short and one long duration symbol for each of the four vowel clusters. The duration of each vowel is determined by the number of successive time slices with the same value of vowel cluster. If the duration is longer than the average duration across the query, then that vowel is labeled long-duration; otherwise it is a short-duration vowel.

In our proposed system, one substantial modification to the traditional edit distance calculation is the choice of penalties that vary for each character in the string. Under the standard edit distance calculation, there is some fixed penalty associated with insertion, deletion, and replacement operations, and these penalties are often equal to unity. Here, we modify the penalty functions to reflect our confidence in the vowel classification that has been assigned. Each character in a string is made up of a number of time slices, all of which must have the same vowel classification. We find a mean spectrum $\bar{\mathbf{x}}$ for each character in the string by averaging the spectra $\{\mathbf{x}_k\}$ from that character's time slices. The character's mean spectrum (the mean across the duration of that character) can then be compared to the vowel cluster's mean spectrum (the mean across *all* time slices that have been identified as that vowel by the NMRA algorithm). If the character spectrum is very similar to the cluster spectrum, we associate a high confidence value with this character identification, and we penalize heavily for any modification to this character. Conversely, if the character spectrum and the cluster spectrum are very dissimilar, then the confidence value should be low, and the penalty for manipulating this character should be slight.

To compute the penalty function for each character, we first find the norm of the difference between the character spectrum and the cluster spectrum in decibels as

$$d = \sum_{k=1}^{K} (\bar{x}_k - \bar{v}_k)^2 \tag{5}$$

where $\bar{x}_i$ is the ith element of the character's mean spectrum and $\bar{v}_i$ is the ith element of the mean cluster vector of the cluster that the character belongs to. We then take the inverse of this value to be the penalty and normalize so that the maximum penalty for any given character string is equal to unity. This penalty function is used for deletion and insertion operations. For a replacement operation, we add the cost of a desired insertion and deletion pair in "quadrature". That is, the replacement of a character with penalty p with another with penalty q yields an overall cost of $\sqrt{p^2 + q^2}$. If we were to add the two penalties directly, then replacement would never be used, as it would be no less expensive than a deletion-insertion pair. By adding the penalties in quadrature, this process becomes analogous to the calculation of error when adding two uncorrelated variables, each of which has an associated error.

2.6. Evaluating the success of the search

In our limited test data set, we have no "correct" vowel stream for each song to use in the evaluation of the string matching success. Instead, we compare every string in the query set to every other string in the query set. Then, we find the nearest neighbor across all queries. We say that a "correct" match occurs when a query from the same song is chosen as the nearest neighbor.

3. Pilot study results

3.1. Vowel clustering

Figure 1 shows a sample data clustering result for a single singer. The four clusters found from the NMRA algorithm are shown in the figure by their respective mean spectra, averaging the spectra (in dB units) across all members of each cluster. As can be seen in the figure, the four spectra are quite distinct and reflect the general properties expected of each of the vowel classes. For instance, the noise/silence spectrum (the heavy solid line) is relatively flat, and the /i/-like spectrum (the light solid line) has a strong second formant, in the 2200 Hz range for this particular singer.

The four spectra above are representative only of a single singer. Each singer's vowel clusters have distinct spectra, as is expected for the different voices. The key is that the different voices all have a shared characteristic—their signals divide into distinct vowel

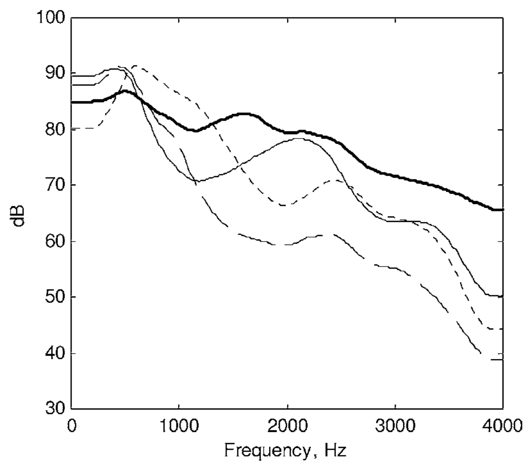

Figure 1. Sample vowel clustering for singer 6 singing a version of "America the Beautiful". Each vowel cluster is shown by its average interpolated spectrum. The heavy solid line is cluster 1 (consonants/silence), the light solid line is cluster 2 (/i/-like), the dotted line is cluster 3 (neutrals), and the dashed line is cluster 4 (/u/-like).

classes. Again, the vowel spectra themselves are not compared across singer, but rather a generalized representation (the stream of vowel labels) is compared. This effectively divorces any idiosyncratic vowel production from the comparison process.

3.2. *Nearest-neighbor matching*

Figure 2 shows the percent correct in the nearest neighbor matches as a function of query song. For each query, the nearest neighbor is found as the vowel stream that requires the

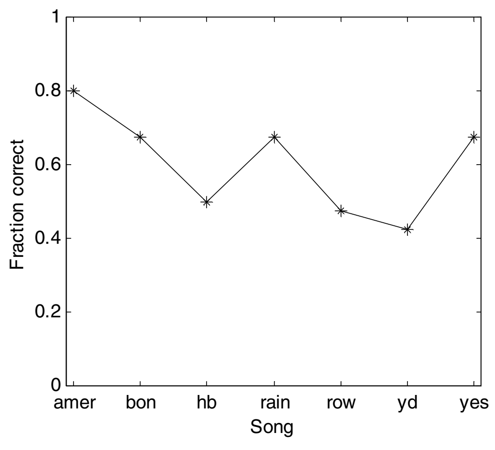

Figure 2. Fraction of correct nearest-neighbor choices as a function of the query song. A correct choice means that the nearest neighbor to the vowel stream is a vowel stream from the same song. There are 40 queries for each song.

fewest number of (weighted) edits to be transformed into the desired query. If that selected vowel stream is generated from the same song as the query, then the selection is deemed correct. There are a total of 40 queries for each song, for a total of 279 possible neighbors from which to select, as we do not count the query itself as a possible neighbor. Therefore, a 14% chance exists of a correct answer occurring at random. Across all songs, the nearest neighbor selection is correctly chosen from the same song 60% of the time, and varies by song between 42% (for "Yankee Doodle") and 80% (for "America the Beautiful").

One parameter that is likely to be influencing the results is the length of the overall strings that are being compared. Since we are performing sub-string matching, there is no penalty for deleting characters at the beginning or ending of the database string. However, there are always penalties incurred for deleting characters in the query string that is being evaluated. In the case of long-duration query strings, the overall edit-distance scores are likely to be larger, because more deletions would have to be made to fit to a shorter string. Thus, long strings are likely to be "closer" to other long strings simply because longer database strings require fewer deletions.

Figure 3 shows the average string length as a function of song, along with the standard deviation. It can be seen that the first song, "America the Beautiful", has the longest average string length. This song is also correctly identified the most often. In general, the trends in figure 3 follow those of figure 3, indicating that string length may be an important factor in the string selection. Indeed, when looking at the incorrect identifications, the overwhelming incorrect choice is "America the Beautiful", containing the longest string.

Figure 4 shows the nearest-neighbor results, broken down by singer instead of by song. The overall average is 60% correct, with the variation by singer ranging between 32% and 100% correct.

It is clear that these two variables, song and singer, have a strong influence over the results of the vowel query. In particular, it is important to note that the four singers with

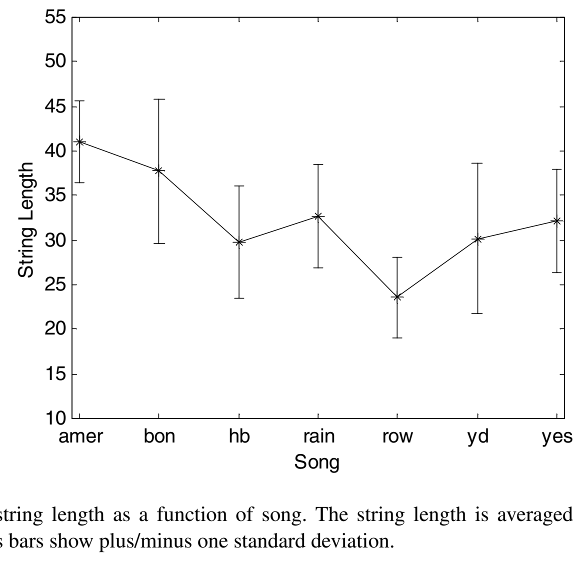

Figure 3. Average string length as a function of song. The string length is averaged across all singers and repetitions. The errors bars show plus/minus one standard deviation.

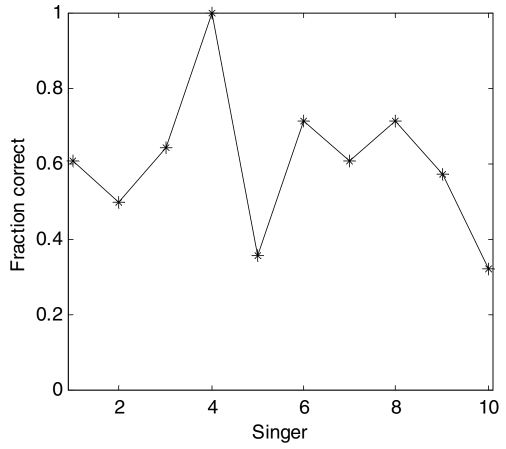

Figure 4. Fraction of correct nearest-neighbor choices as a function of the query singer. A correct choice means that the nearest neighbor to the vowel stream is a vowel stream from the same song. There are 28 queries for each singer.

no prior musical experience were singers 2, 5, 8, and 10. The vowels from three of these singers (2, 5, and 10) were the most poorly identified of all the singers. This indicates that musical training may influence vowel production, which is a less obvious connection than its influence on pitch and rhythm production.

Figure 5 shows the fraction of queries for which the nearest neighbor is sung by the same singer, as a function of the singer number. In general, the fraction of nearest neighbors from the same singer corresponds roughly to the fraction correctly identified for each singer. For singer 4, whose queries were correctly identified 100% of the time, many (79%) of the selected queries were sung by that same singer. For singer 10, whose queries were

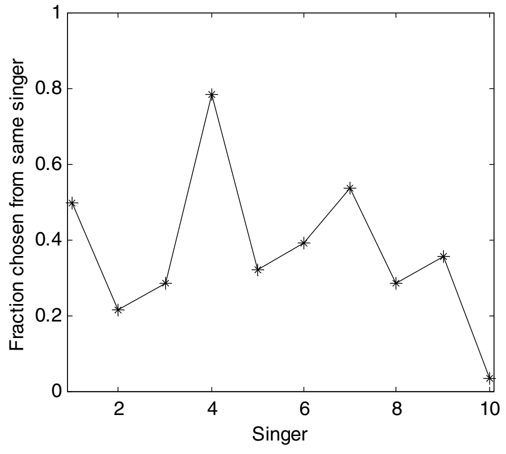

Figure 5. Fraction of nearest-neighbor choices that are sung by the same singer as the query, as a function of singer (regardless whether than choice was correct).

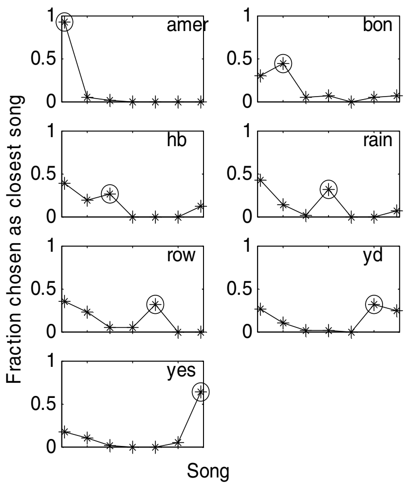

Figure 6. Each panel shows the results for the 40 queries that originate from the song labeled in the top-right corner of the panel. For each query, the average distance to each song is computed, averaging across the queries from that song. The shortest average distance is selected between each query-song pair. The panels show the fraction of time each song was selected as having the shortest average distance for that panel's queries. The song that is circled is the correct response, corresponding to the number of times the query song was selected as being the best choice.

identified least, almost none of the selected queries were sung by singer 10. Again, the trends in figure 5 seem to mirror those of figure 4, indicating that the matching is influenced not by the singer of the given test query, but also by the singer of the reference queries.

3.3. Average distances between queries and songs

The nearest neighbor measure can be prone to outlier behavior. For instance, we may be able to randomly generate 280 character strings and find one that closely matches a given query. Rather than working with isolated matches, we can also look at average matches across a population of character strings. For each song, we have 40 sung instances. We can then measure the average distance between each query and the 39 remaining instances of that song, as well as the average distance between each query and the 40 instances of each of the remaining songs. Ideally, the average distance between a query and its song should be shorter than the average distance between a query and all other songs.

Figure 6 shows the average distance between queries and songs. Each panel represents the queries from a different song, so each panel represents the results from 40 queries. The x-axis shows the different songs, and the y-axis shows the fraction of the 40 queries for which the x-axis song was chosen as being the closest, after averaging the distances across all queries corresponding to that song. For instance, looking at the "Happy Birthday" panel, of the 40 "Happy Birthday" queries, roughly 40% them were closest to the "America" character strings, roughly 20% were closest to the "My Bonnie" strings, etc. In each panel, the correct song is circled. If the results were 100% correct, then we should see a delta function at 100% for the circled element, and all other elements would be at 0%. On the other hand, if the results were randomly selected, then we should see a more-or-less flat line across all songs, with each one receiving roughly 14% of the total.

We can see from the panels in figure 6 that four of the seven songs were correctly chosen as being closest to the majority of their queries. The remaining three had the correct song as the second choice. It is interesting to note that for the three second-choice results, the first-choice result was always "America the Beautiful", which correlates with the fact that the string lengths for the "America" character strings are somewhat longer than those of the other songs. Query length may influence the matching results: it is more likely that a sub-string of a long reference string can be found to match a given query than that of a short reference string.

4. System analysis

4.1. Qualitative analysis

The results of this small-scale study illustrate the potential viability of vowel-stream based music information retrieval. However, the system presented here is far from perfect. The system is currently only 60% accurate on a database of only seven songs. This suggests that the algorithm as presented may not scale well to larger databases. In this section, the system is analyzed to identify potential areas of improvement.

Currently, the weakest element of the system is the vowel extraction and classification. It is not surprising that this task is difficult given the sizable body of research devoted to unconstrained speech recognition. The four-cluster model was chosen to reduce the complexity of the task to manageable levels, and it is remarkable that the NMRA clustering algorithm naturally tends to select clusters that roughly correspond to non-vocalic sounds and front, neutral, and back vowels. However, an analysis of the resulting clusters shows that the simple approach taken here is not sufficient to produce robust and consistent classification of vowels. This can be seen most clearly by examining the results of the thirty NMRA trials run on each query. While many of the queries exhibit relatively little variation with only one clustering solution strongly evident, a significant number of queries show two or more sets of clusters that were each selected many times by the NMRA algorithm. In most of these cases, such a lack of cluster robustness will set off the third warning test, which indicates that 20% of the cluster solutions differ from the mode. Unfortunately, this indicates a shortcoming in the analysis procedure rather than in the query itself.

Another problematic feature of the NMRA clustering results is the inconsistency of clusters between different repetitions of the same query by the same singer. One might reasonably expect clusters to vary somewhat between singers or even among the different songs being queried for the same singer. However, an inherent assumption of a query system is that repeated queries should be represented by the system as roughly equivalent. Unfortunately, the NMRA algorithm does not always represent the vowel streams for these repeated queries in the same way. In some cases, the errors may be caused by improper cluster registration across queries. There is some evidence that sorting the cluster means by spectral moment may not be sufficient to register the clusters in all cases, particularly for higher pitched voices and voices that exhibit a strong singer's formant. In most cases, though, the variations for queries of the same song by the same singer cannot be removed by a simple permutation of the cluster labels.

Ultimately, the problems with the NMRA-based vowel classifications seem to be primarily a result of the use of an unsupervised classification method on data that does not exhibit clearly defined clusters. This may be partly dependent upon the spectral representation used for classification, but so far experiments have shown that the particular representation used has little effect on the final results. For instance, the use of more traditional (and compact) representations such as mel-frequency cepstral coefficients (Rabiner and Juang, 1993) produces little change in the system's overall performance. While it is possible that representations exist that may provide sufficient class separation to improve the performance of the system, the nature of the data seems to suggest otherwise. For queries sung in English and many other languages, vowels are not pure monophthongs. Thus, a given vowel may transition smoothly from a front vowel to a neutral one, or from a back vowel to a front one. Any frame-by-frame classifier will have difficulty with this type of data, since it will necessarily need to classify frames that are "in the middle" between two classes. This suggests the need for some higher-level analysis of the spectral data or clustering results either before, during, or after clustering to improve cluster robustness.

The general instability of the NMRA algorithm also suggests that the string representation employed by this system may not be ideal. While such strings might be a fine representation given a relatively error-free representation of the vowel stream, string representations tends to exacerbate the instabilities inherent in the clustering algorithm. Suppose an extended vowel falls on the border between two of the clusters so that it is alternately classified as one and then the other. The resulting string will be populated by numerous short vowels rather than a single long one. When compared to a string for which this vowel was properly classified, there will be a significant edit distance penalty. The difficulties introduced by this situation are alleviated somewhat by the classification confidence scores used in the edit distance calculations. Unfortunately, a similar and more common problem exists that is not directly addressed by this confidence measure. Typically, short vowels interspersed with consonants (as often occur during a passage of rapid notes) are not classified consistently. The result is a possibly lengthy string of short vowels which may have little to do with the vowels actually sung. Ideally, the system would treat these segments as suspect and reduce their influence on query comparisons. In this system, however, rapid passages such as this may be encoded as lengthy portions of the string, owing to the rapid spectral changes that occur during such portions of the query.

Unfortunately, it is not clear what form a better representation might take. Intuitively, one might suggest that duration should play a more key role in such a representation. This takes advantage of the "landmark" property of long, extended vowels, which mark long notes, typically at the ends of phrases. Visually, the use of such "landmark vowels" seems to be the easiest way to register two similar vowel streams. This also exploits the increased likelihood that extended frames of similarly classified data have been correctly classified. Such a duration confidence score might be employed instead of or in addition to the classification confidence currently employed.

4.2. Edit distance analysis

One of the most prominent effects noted in the results of the present study is the strong correlation between string length and retrieval performance. This effect can be explained as a result of the sub-string matching methodology used. When a query string is longer that the database string it is compared against, a penalty to the edit distance based on the difference in length between the two strings is incurred. If, however, the query string is shorter than the database string, the edit distance will actually be reduced because of the greater number of unpenalized edit sequences that may be matched. A Monte Carlo experiment was employed to examine this phenomenon quantitatively. Two uniformly random 8-symbol strings with a particular length ratio were generated and the edit distance between the two strings was measured. For simplicity, the confidence penalty was neglected so that each string edit operation was assigned unity cost. For each length ratio, this experiment was repeated 10,000 times. The edit distances are normalized by the length of the query string. The mean and standard deviation of the normalized edit distances are plotted as a function of the ratio between the database string and the query string in figure 7. The string length effects noted above can be clearly seen in this figure.

Figure 7 shows that longer strings in the database will tend to have shorter edit distances to a given string than will short strings. Thus, it is not surprising that longer strings would have a higher probability of selection. Since the present study also shows this trend, we examine the actual edit distances between strings used in the present study. Figure 8 shows the average normalized edit distances from our study plotted as a function of string length

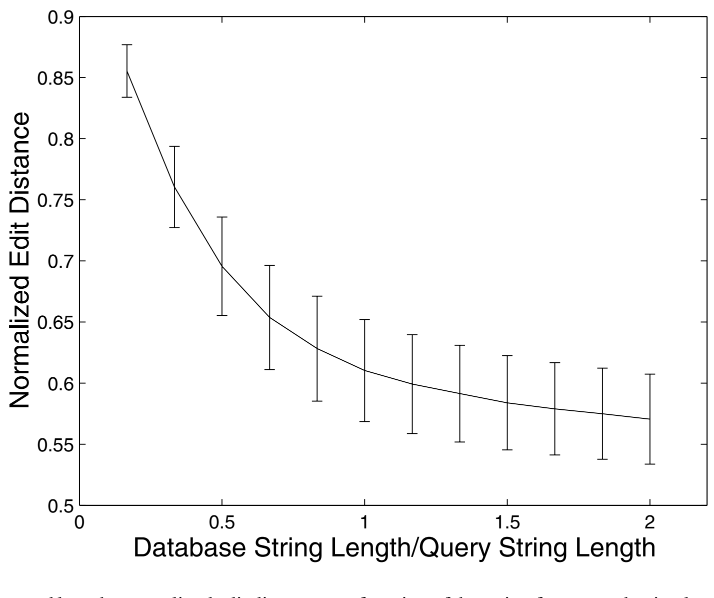

Figure 7. Expected length-normalized edit distance as a function of the ratio of compared string lengths, measured using Monte Carlo simulation. The edit distance is normalized by the length of the query string. Error bars indicate plus/minus one standard deviation.

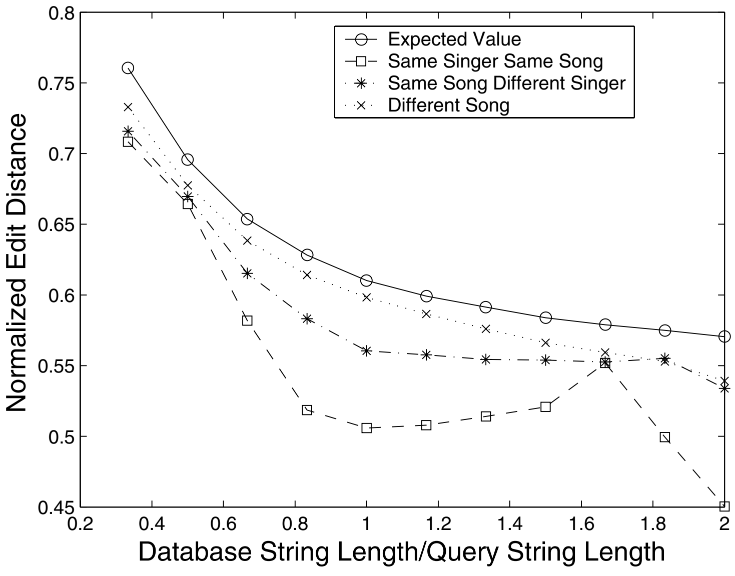

Figure 8. Average length-normalized edit distances between strings used in the pilot study as a function of the ratio of compared string lengths. The various classes of comparison (same song same singer, same song different singer, and different song) are shown separately.

ratio. In this figure, the data is separated into three classes: "same song/same singer," "same song/different singer," and "different song." The expected normalized edit distance is also plotted for comparison.

We expect the different-song class to behave in a manner similar to the expected normalized edit distance. Figure 8 shows that the different-song class has a slightly lower edit distance than the expected value. This is most likely due to correlations between the different songs or the production mechanism that are not accounted for by the uniform-random-string model. We further expect that the same-song classes will have lower normalized edit distances than the different-song class, with the same-singer class having the lowest normalized edit distance. Figure 8 does in fact show this trend; however, there is not as much separation between the classes as would be ideal. It should be noted that the classes have similar standard deviations to those shown in figure 7; the standard deviations are not plotted for clarity. This means that the mean value of each class is generally within one to two standard deviations of the others. This overlap in edit distance between classes reduces retrieval performance significantly.

We can see in figure 8 that the same-song classes are not exclusively dominated by the trend shown in the different-song class and the expected normalized edit distance. For the same-song classes, there is local minimum at the string-length-ratio of 1. This indicates correlation between strings generated by same-song queries, and suggests that a different model for same-song error should be employed. One potential alternative model suggests that each query string is an error-corrupted realization of some ideal string. Then, the edit distance between two same-song queries can be attributed to an error insertion mechanism.

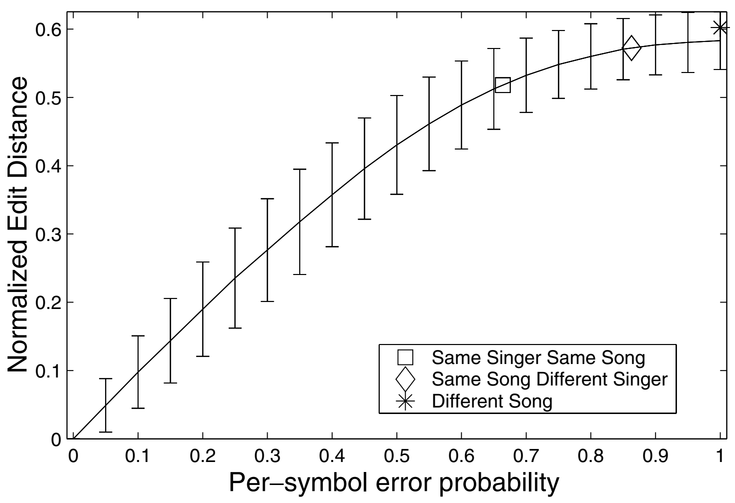

Figure 9. Expected normalized edit distance between a string and its corrupted version plotted as a function of per-symbol error probability. Bars indicate plus/minus one standard deviation of the normalized edit distance. The symbols show the intersection of this curve with the mean normalized edit distance obtained for each of three comparison classes.

Under such a model, we expect deviations from a length ratio of one to result in higher edit distances. Then, as error begins to dominate the string, the trend seen in figure 7 will become more prominent. For simplicity, we model the error as a uniform per-symbol probability of error, which may be an insertion, deletion, or symbol replacement (with equal probability of each type of error). By measuring the edit distances resulting from such an error insertion process, we can estimate the error probability that links different strings in each of the classes.

Figure 9 shows the expected normalized edit distance as a function of the per-symbol probability of error. Under this model, changes in length result from the error insertion process and thus length is not treated as an independent variable. Symbols are included to indicate the intersection of each class with the expected edit distance curve. For the same-singer/same-song and same-singer/different-song classes, the mean normalized edit distances are 0.518 and 0.574, respectively. These edit distances correspond to equivalent per-symbol error probabilities of 0.66 and 0.86. Note that the expectation curve does not reach the mean normalized edit distance of 0.602 for the different-song class below an error probability of 1. This is because of our error insertion model, under which an error probability of 1 does not completely decorrelate the corrupted string from the original string. The equivalent per-symbol error probabilities for the other two classes are very high—especially since one would hope that two queries by the same singer would be represented as similar strings.

One particularly interesting feature of figure 9 is the decrease in slope at large values of per-symbol error probability, which indicates that, at these values, different classes of comparison are less separated by edit distance than at smaller values of per-symbol error probability. On the one hand, this shows that the sub-string-based edit distance employed here loses its ability to distinguish classes as queries become more strongly corrupted by error. This is particularly true for the current system, where the equivalent per-symbol error probabilities are significant. On the other hand, this suggests that by reducing the variability of the string coding methodology, we should be able to significantly improve class separation and thus retrieval performance. Further, this indicates that each incremental decrease in error will yield a greater increase in class separability.

5. Discussion

The results of the present small-scale study indicate that the vowel stream produced in a sung query does provide sufficient information to allow for database retrieval. Further, we have seen that enough differentiation exists between the front, neutral, and back vowels of a singer to separate and crudely identify them using only a simple clustering algorithm. The vowel stream may form an important part of a music retrieval system, especially for queries that may contain little useful pitch or rhythmic information. However, before the system can become generally useful a number of improvements need to be made.

The most important improvement that must be made to the system involves the vowel classification subsystem. The NMRA vowel classification works well enough on individual queries, but lacks consistency across queries that should be very similar. This is partly a result of the use of an analysis method that is entirely self-referential in nature. The use of a self-referential method of analysis and classification provides many benefits, such as the ability to compare singers directly regardless of the various factors that alter vowel quality and the ability to process a query without training. Unfortunately, the use of a purely self-referential analysis makes the classification task more difficult and tends to degrade consistency across queries. The introduction of a short training phase employing a well-chosen set of lyrics and system-guided rhythmic alignment could improve the consistency of classification results while also allowing the collection of useful statistics about vowel production. The resulting analysis could employ both self-reference and other cross-query information to perform classification.

Another possible improvement could come by performing classification at a higher level than frame-by-frame. Using similarity matrices like those investigated by Foote (1999) and Bartsch and Wakefield (2001), one can observe extended blocks of frame similarity that might be grouped together for classification to eliminate spurious cluster transitions. This may also improve classification accuracy. One of the primary difficulties with this approach would be dealing with diphthongs and fast vowel sequences. This analysis might also examine vowel prominence and stability for the purpose of detecting "vowel landmarks" that could form the basis of a somewhat different representation of vowel streams.

Ultimately, it will also be necessary to find some method of representing vowel streams so that the search database can be populated with something other than queries. It is conceivable that useful vowel streams could be generated in a manner analogous to text-to-speech or text-to-singing synthesis. In this manner, one could generate an entire database of vowel streams from a database of lyrics or lyrically annotated musical scores. Ideally, the vowel streams would be extracted from original recordings when available; however this task is likely to be roughly as difficult as general-purpose melodic extraction. The tradeoff is very similar to that made by many music retrieval systems that employ symbolic musical representations rather than attempt to work with audio directly.

6. Conclusions

This study suggests that a simple clustering of the spectral content of a sung query can isolate different vowel classes within a single query. When applied to a small-scale database of 40 instances of 7 songs, 60% of the queries were correctly identified using only the vowel stream. We expect that the addition of pitch and rhythm will substantially increase this performance, and that vowel identification can be used to complement other methods of music information retrieval.

Acknowledgments

This work is supported, in part, by the National Science Foundation Information Technology Research initiative and also by a National Science Foundation Graduate Research Fellowship granted to the second author. We thank our collaborators, particularly William Birmingham and Roger Dannenberg, for their input in the music information retrieval work, and to Muriel Makarim for her assistance with data analysis. We also acknowledge all of the volunteers who sang for us.

References

Bartsch, M.A. and Wakefield, G.H. (2001). To Catch a Chorus: Using Chroma-based Representations for Audio Thumbnailing. In *Proceedings of the 2001 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, (pp. 15–18). New Paltz, NY.

Birmingham, W.P., Dannenberg, R.D., Wakefield, G.H., Bartsch, M.A., Bykowski, D., Mazzoni, D., Meek, C., Mellody, M., and Rand, B. (2001). MUSART: Music retrieval via aural queries. In *Proceedings of ISMIR 2001: 2nd Annual International Symposium on Music Information Retrieval*. (pp. 73–82). Bloomington, Indiana University.

Foote, J. (1999). Automatic Audio Segmentation using a Measure of Audio Novelty. In *Proceedings of IEEE International Conference on Multimedia and Expo*, I, (pp. 452–455).

Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition*, 2nd edn., San Diego, CA: Academic Press.

Ghias, A., Logan, J., Chamberlin, D., and Smith, B.C. (1995). Query By Humming: Musical Information Retrieval in an Audio Database. In *Proceedings Proceedings of the Third ACM International Conference on Multimedia*, (pp. 231–236).

Hermes, D. (1993). Pitch Analysis. In M. Cooke, S. Beet, and M. Crawford (Eds.), *Visual Representations of Speech Signals* (pp. 3–25), New York, NY: Wiley & Sons, Inc.

Mazzoni, D. and Dannenberg, R.D. (2001). Melody Matching Directly from Audio. In *Proceedings of ISMIR 2001: 2nd Annual International Symposium on Music Information Retrieval*, (pp. 73–82). Bloomington: Indiana University.

McNab, R.J., Smith, L.A., and Witten, I.H. (1996). Signal Processing for Melody Transcription. In *Proceedings of ACSC'96: Nineteenth Australasian Computer Science Conference*, (pp. 301–307). Melbourne, Australia.

McNab, R.J., Smith, L.A., Witten, I.H., and Henderson, C.L. (2000). Tune Retrieval in the Multimedia Library. In *Multimedia Tools and Applications*, 2/3, (pp. 113–133).

Meek, C. and Birmingham, W.P. (2002). Johnny Can't Sing: A Comprehensive Error Model for Sung Music Queries. In *Proceedings of ISMIR 2002: 3rd International Conference on Music Information Retrieval*, Paris, France.

Miller, R. (1996). *On the Art of Singing*, (p. 50), New York, NY: Oxford University Press.

Needleman, S.B. and Wunsch, C.D. (1970). A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins. *J. Mol. Biol.*, 48, (pp. 443–453).

Rabiner, L.R. and Juang, B.-H. (1993). *Fundamentals of Speech Recognition*, Englewood Cliffs, NJ: Prentice-Hall, Inc.

Rabiner, L.R. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proc. of the IEEE*, 77(2), (pp. 257–286).

Shifrin, J., Pardo, B., Meek, C., and Birmingham, W.P. (2002). HMM-Based Musical Query Retrieval. In *Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries* , (pp. 295–300). Portland, OR.

Smith, L.A., McNab, R.J., and Witten, I.H. (1997). Music Information Retrieval Using Audio Input. In *Proceedings of the AAAI: Intelligent Integration and Use of Test, Image, Video and Audio Corpora*, (pp. 12–16). Stanford, CA.

Titze, I. (1994). *Principles of Voice Production*, Prentice-Hall, Inc., Englewood Cliffs, NJ.