Diatonic Key Finding in a Query by Humming System

Thomas Gersic
Northwestern University
http://www.gersic.com

ABSTRACT

I describe an extension to the melody matching system used in Gersic [7]. The system takes user input in the form of a sung or hummed query recorded through a microphone. Once recorded, the query is transcribed into a string of quantized MIDI notes which are matched to a series of similarly transcribed audio files. The matching process is executed by means of a local-alignment edit distance algorithm. The extension described here leaves the system intact, but adds a diatonic key-finding system between the transcription and matching processes. Once the diatonic key of a query has been discerned, this information can be used to reduce transcription or singing errors.

1. INTRODUCTION

The sale of music on the Internet is a multi-million dollar industry which has made it possible for a vast array of music to be available to the average household without the buyer having to leave the comfort of their home. The iTunes music store alone offers more than one million songs available for download, and the ability to quickly and accurately find a song is important to the business model. Searching through one million songs can be easy if the user knows the name of the song or the artist, but can be a daunting task if this information is not known. Birmingham et al [1] contend that music search engines are hindered by their use of meta-information for searches instead of musical content, which is the information with which consumers are most familiar.

To hear a song on the radio and be able to sing that song into a microphone in order to identify and purchase it could make for a significant increase in sales of music online. However, the quality of current query-by-humming systems fall somewhat short of being useful in a commercial environment. A commercially viable system must be able to quickly search a large database of songs, and contend with a wide variety of vocal performance abilities. If a user of a query by humming system sings perfectly on pitch into a high-quality microphone, they have a much better chance of getting good results. However, most users are more likely to do a poor job of singing on pitch, and will sing into a low-quality microphone. Dealing with the issue of query-quality is a necessary task for developing a system which is useable on a large scale. I demonstrate a system which attempts to detect a diatonic key for the query, and to subsequently use that key information to modify the transcription in order to remove likely pitch errors.

2. SYSTEM DESCRIPTION

The system I describe is in five parts: the data set, the transcription system, the key-finding system, the key-based error reduction system, and the matching system. The data set consists of stored monophonic recordings of main themes for popular songs. This data set is queried by a user, who sings or hums a melody into a microphone. The user's query is transcribed into a series of frequencies by the praat-pd algorithm [2], and these frequencies are then quantized both in time and in frequency into a series of MIDI pitches according to the rules of equal temperament. These MIDI pitches are then analyzed to find a diatonic key, and the key information is used to modify the transcription so that transient out of key pitches are moved to fit into the key.

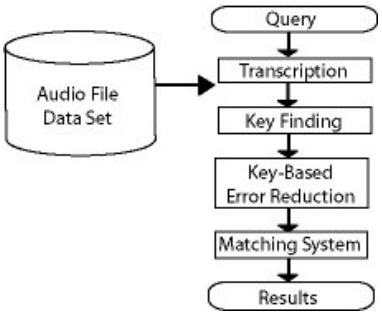

Figure 1. Music Matching System Diagram

Once the transcription has been modified, it is matched against transcriptions of stored musical motives in an effort to find the stored file most similar to the query file.

2.1 Pitch Tracking and Quantization

Pitch Tracking was accomplished through the use of the praat-pd algorithm, coded by Mark Bartsch, modified by Bryan Pardo, and based on praat by Paul Boersma [2]. The algorithm works by windowing a signal in the time domain in order to determine the period and harmonicity (harmonics to noise ratio) of a sound. Boersma shows this method to have an accuracy and reliability greater than that of frequency-domain methods, and to return good results for sounds up to 80% of the Nyquist frequency. Specific input settings for the praat-pd algorithm are the same as in Gersic [7].

The praat-pd algorithm returns a list of frequencies for each window, so in order to limit the total amount of data, frequencies are quantized to MIDI pitches. A simple formula for converting from frequency to MIDI pitch is used in accordance to the rules of equal temperament. Additionally, the series of pitches is quantized in time by detecting the attack transients of each note sung. To do this, the discrete derivatives of the praat-pd frequency path and the autocorrelation (harmonicity) path are calculated. New notes occur whenever the frequency or harmonicity changes positively to a greater degree than the average change in frequency or harmonicity throughout the transcription. This method works better with sung recordings, as opposed to hummed, because the attack transients caused by the consonance of each word make much larger changes in frequency and harmonicity. Once starting points are determined for each note, the average of the frequencies between starting points is used to determine the frequency for each note.

2.2 Edit-distance Alignment

The edit-distance alignment algorithm is a method taken from Durbin et al [4]. This method, shown by Uitdenbogerd and Zobel [10] to be effective and efficient for melodic theme matching, compares two strings in an attempt to determine how similar they are to each other by calculating the number of substitutions necessary to change one string into the other. The edit-distance algorithm generates a matrix of data that can be used to determine either a global-alignment value or a local-alignment value. Local-alignment values, with a match reward of 15 and a skip cost of 2 were found to give the best results in Gersic [7], and are used in all trials here. With this algorithm, higher numbers indicate better matches, and values are normalized to a range between 0.0 and 1.0 (1.0 is a perfect match). Normalization is accomplished by dividing the match score by the product of the size of the shorter of the comparison strings and the match reward setting.

2.3 Krumhansl & Schmuckler Key-finding

The Krumhansl and Schmuckler key-finding algorithm [8] [5] is based on key profiles established through empirical work by Krumhansl and Kessler [9]. The key profiles were obtained through a sequence of experiments where a tonal center was established by playing an incomplete diatonic scale or chord sequence, and followed in turn by each of the twelve chromatic pitches. Participants were asked to rate how well each pitch fit the established tonal center. In the algorithm, key profiles are correlated with the pitch-class distribution of the transcriptions. Each pitch-class is weighted by its duration throughout the recording, and the algorithm returns the diatonic key with the highest correlation rating. The algorithm has the effect of returning keys based on the relative popularity of certain pitches throughout the transcription. For instance, for most of western music, if a piece is in the key of C, the most popular pitch-class will be C, followed by G and E. Other scalar pitches (D, F, A, and B) follow, and the least likely pitches to occur are ones that fall outside of the key, such as Db, Eb, or Ab.

2.4 Spiral Array CEG Key-finding

The Spiral Array Center of Effect Generator (CEG) algorithm [3] [6] is an alternate key-finding algorithm which has been shown to return more accurate results than the Krumhansl and Schmuckler method. Prior to key-finding, a set of key-profiles are determined for each of the 24 major and minor keys. Notes are arranged around a three-dimensional spiral in ascending perfect fifths so that major-thirds are vertical neighbors, and perfect fifths are horizontal neighbors around the spiral. Key profiles are determined by finding the center of effect for each of the tonic, subdominant, and dominant chords. Those three centers of effect are then used to find the center of effect of the key. Once these profiles have been determined, the key of a selection is computed by plotting pitch-classes along the spiral, weighting them by the cumulative duration for each class, and finding the center of effect. The key profile with the closest center of effect to the selection's center of effect is returned as the key of the selection.

2.5 Human Key-finding

To provide for a basis of comparison, and to evaluate the performance of the Krumhansl and Schmuckler and Spiral Array CEG algorithm apart from the performance of the key-based error reduction system, each of the recordings in the corpora were analyzed by a semi-professional musician in order to determine how well the system works with an alternate key finding mechanism. The musician, who is also the author, has more than twenty years of formal musical training. Each piece in the corpora was transcribed by hand, and diatonic keys were chosen and stored in a lookup table for use by the key-finding system. This method had the result of returning much more consistent keys for alternate recordings of pieces than did the Krumhansl and Schmuckler or the Spiral Array CEG algorithm. One likely reason for this is that the transcribed selections were short, and since they were monophonic, they contained too little harmonic information for the algorithms to work optimally.

Table 1. Comparison Between Krumhansl & Schmuckler Algorithm, Spiral Array, and Human Key-Finding

Song	K&S	Spiral	Human
99 Red Balloons v1 words	F Minor	C Major	F Major
99 Red Balloons v1 nowords	C# Major	F Minor	F Major
99 Red Balloons v2 words	E Major	D# Major	F Major
99 Red Balloons v2 nowords	D Major	A# Major	F Major

2.6 Diatonic Key-based Error Reduction

Once the key of a selection is determined, it is necessary to apply the key-profile to the transcription in order to reduce errors caused by incorrectly sung pitches. To do this, major and minor key profiles are defined by interval relations. The tonic of the scale is set equal to the key determined previously during key-finding. Further steps of the scale are defined by adding intervals to the tonic. Pitch-classes are arbitrarily assigned a number, where C is equal to 1, C# is equal to 2, etc.. If the selection is in the key of E, then the tonic is set to 5, and the next step of the major or minor scale is 5 plus 2, which is F#. The rest of the key profile is defined in this manner. After the key-profile is determined, it is compared with the transcription in order to find pitches that are outside of the key. While western music does allow for pitches which are out of key, they are relatively less common than in-key pitches, so they are assumed to be more likely to have been sung or transcribed incorrectly. Two different error-reduction techniques are described.

2.5.1 Neighbor-Pitch Substitution

As each pitch is considered in turn, any out-of key pitches are replaced by either the preceding in-key pitch, or if the pitch currently being considered is the first pitch, the closest pitch, as defined in section 2.5.2.

2.5.2 Closest-Pitch Substitution

Out-of-key pitches are analyzed in order to determine if they were sung flat or sharp. Since pitches are quantized to MIDI notes before comparison, tuning information is lost. For instance, if a user sings a middle C, the system will represent it as MIDI note 60. However, if the pitch were not quantized when converted from frequency, and were sung sharp, it may actually have been closer to 60.4. If a sung-pitch is determined to be out of key and sharp, that pitch is replaced by the next pitch up the chromatic scale. If a sung-pitch is determined to be out of key and flat, that pitch is replaced by the next pitch down the chromatic scale. Since an out-of-key pitch is only one half-step away from being in-key in either direction of a diatonic scale, moving it one half-step upwards or downwards ensures that it will become an in-key pitch.

It was hypothesized that limiting the number of pitches available to be replaced would result in better performance. This takes into account the possibility that the diatonic key could have been determined incorrectly, or that a short modulation may have occurred in the music. It is important to distinguish between intentional and accidental out-of-key pitches, so two different limit criteria are tested. The first is based on the duration of the pitch in question, when compared with the average duration of pitches in the selection. Only pitches less than one-fourth or one-eighth (depending on the trial) the duration of the average are replaced. The second limit criteria is the degree to which the pitch is sharp or flat. Since the quantization process rounds the results of the conversion from frequency to MIDI pitch, a pitch can be up to 50% sharp or 50% flat without being considered a different pitch by the transcription system.

3. EXPERIMENTAL SECTION

Three different key-finding methods are tested and compared. In addition, error-reduction techniques are compared.

3.1 Corpora Construction

Two corpora were used during experimentation. Corpus A is exactly the same as is described in Gersic [7]. It consists of forty recordings of four different pieces, each recorded five times with words and five times without words. The four pieces are: *99 Red Balloons*; the *Gilligan's Island* theme song; *Allouette*; and *Jingle Bell Rock*. *99 Red Balloons* and *Gilligan's Island* were recorded by a 26 year old male with a Labtec AM-240 unidirectional electret microphone with a frequency response from 100 to 16000 Hz. These songs were recorded with Matlab 7.0 (R14), and linearly encoded as a PCM waveform with a sampling rate of 11025 Hz, and a bit depth of 16 bit. *Allouette* and *Jingle Bell Rock* were recorded by a male in his late 30's. These songs were also linearly encoded as a PCM waveform, with a sampling rate of 8000 HZ, and a bit depth of 16 bit.

Corpus B extends and includes Corpus A. It consists of one-hundred recordings of ten different pieces. Each piece is recorded five times with words and five times without. In addition to the forty recordings from Corpus A, Corpus B also contains *My Favorite Things*, *Let it Be, Love Me Tender, My Girl, Stand by Me,* and *We Three Kings of Orient Are.* The additional pieces were sung into a Shure SM-57 microphone by a 26 year old female with a BA in vocal education. They were recorded with Syntrillium Cool Edit, and linearly encoded as a PCM waveform with a sampling rate of 11025 Hz, and a bit depth of 16 bit. The average length of the sound files in Corpus A is 4.81 seconds, and the average length of the sound files in Corpus B is 11.38 seconds. All files were recorded as monaural files.

3.2 Evaluation method

Two different evaluation methods were considered. For each trial a data set was created by comparing each file with every other file in the corpus in order to determine edit distance scores. Since both corpora contained multiple recordings of the same song, sung by the same person, the first evaluation method was to compare the average score generated for matches of songs that were the same versus the average score for songs that were different. Since the scores were quantized to a range from 0 to 1, scores for recordings of the same song should be closer to 1 than scores for different songs. For instance, the similarity rating for *99 Red Balloons: take 1 (with words)* compared to *99 Red Balloons: take 4 (without words)* should be closer to 1 than the similarity rating for *99 Red Balloons: take 1 (with words)* compared to *Jingle Bell Rock: take 2 (with words)*. Ratios of same scores to different scores (S:D) are used as a basis of comparison.

The second evaluation method was to determine whether a user's query to the system would yield a correct answer. A correct answer is defined as the system returning the highest score for a recording of a song that was the same as the query song. For instance, if recording #1 of *99 Red Balloons* was the query song, a correct answer would be recording #4 of *99 Red Balloons*, but an incorrect answer would be recording #2 of *Jingle Bell Rock*. The percentage of correct matches (POCM) is used as a basis of comparison.

3.3 Results

Baseline results for Corpus A are set equal to the best results from Gersic [7]. Optimal edit-distance settings were able to yield a same-to-different (S:D) ratio of 1.732807 and a percentage of correct matches of 85%. Since Corpus B contained more recordings, baseline scores were somewhat lower. Using the same edit-distance settings as with Corpus A, the Corpus B baseline S:D ratio was 1.741765, and the POCM was 68.00%. Results within each corpus which are better than these are considered to be an improvement to the system.

Neighbor-pitch substitution yielded worse than baseline results with all three key-finding methods, and will not be discussed further. Closest-pitch substitution yielded promising results in several trials. The best overall results were obtained on Corpus A, using the human perception lookup table, modifying out-of-key pitches through closest pitch substitution, limited by duration to one-fourth the average duration of the pitches in a transcription. This resulted in a S:D ratio of 0.746953, and a POCM of 90.00%. With Corpus B, the highest S:D ratio of 1.931979 was obtained through using the human key-finding method, closest pitch substitution, with no limits placed on which out-of-key notes are able to be replaced. However, this yielded a POCM lower than baseline, at 67.00%. None of the Corpus B trials were able to yield a higher than baseline POCM. This is potentially the result of the higher quality vocals in the extra Corpus B files, when compared to the Corpus A files. They were sung by a vocalist who was instructed to sing without any vibrato, so the pitch of the extra files in Corpus B is both very accurate and very steady, leaving few errors for the key-based error reduction system to remove. It seems that error reduction is best performed on a data set that contains errors. This is an area for further study.

The use of the Krumhansl and Schmuckler algorithm yielded worse than baseline results in all trials. Human perception results were the best overall, but the Spiral Array algorithm also performed well. With closest pitch substitution, limited by a duration of one-quarter the average duration, the S:D ratio of 1.859529 was the highest of all Corpus A trials. In addition, aside from the human perception trial mentioned earlier, the Spiral Array was the only method able to raise the POCM above baseline. With closest-pitch substitution on Corpus A, limited by degree of intonation, the Spiral Array algorithm raised the POCM to 87.50%.

3.4 Analysis

Of the three key-finding methods tested, human perception scored the highest, while the Krumhansl algorithm scored the lowest. This suggests that the Spiral Array algorithm, in accordance with Chew [9], does indeed produce results closer to human perception with short sequences of notes. To determine why the Spiral Array algorithm returned better results in this system, a statistical analysis was performed on the edit distance scores between the scales of the keys returned for human perception when compared to the Krumhansl and Schmuckler algorithm and for human perception when compared to the Spiral Array algorithm. For instance, if human perception returned a key of F Major, and the Krumhansl and Schmuckler algorithm returned a key of F Minor, the edit distance was calculated for the scales [F, G, A, Bb, C, D, E] and [F, G, Ab, Bb, C, Db, Eb]. Scales with more similar pitch-classes will have higher edit-distance scores. Human perception yielded the best results in this study, so it was used as the basis of comparison between the algorithms. Edit distance input settings were the same as used for comparison between files earlier in the study.

It was hypothesized that the average, normalized edit distance between the Spiral Array algorithm and human perception would be closer to 1 than the average edit distance for the Krumhansl and Schmuckler algorithm. This would indicate that the keys returned by the Spiral Array algorithm were more similar to the keys returned by human perception than those returned by Krumhansl and Schmuckler. Interestingly, this was not the case. The average edit distance for the Spiral Array when compared to human perception was 0.595298, whereas the average edit distance for Krumhansl and Schmuckler when compared to human perception was 0.526548. This seems to suggest that the Krumhansl and Schmuckler algorithm returned keys more similar to human perception than did the Spiral Array algorithm. However, the standard deviation of the keys returned for the Spiral Array algorithm was lower than that of Krumhansl and Schmuckler. This seems to suggest that while the Krumhansl and Schmuckler algorithm returned results more similar to human perception, the Spiral Array algorithm returned more consistent results. It would appear that the query by humming system is aided more by a consistent key-finding algorithm than by one that more closely mimics human perception.

4. CONCLUSIONS AND FUTURE WORK

Several trials yielded promising results for error reduction models based on diatonic key-finding. Human perception yielded the most consistent results, and also the highest increase in POCM score. This shows that error reduction models based on correct key information do indeed raise the accuracy of a query-by-humming system. The accuracy of the Krumhansl and Schmuckler algorithm proved to be insufficient to aid the system, but the Spiral Array CEG algorithm was able to produce good results when the keys being replaced were limited by the degree to which they were out of tune. The best results required that replaceable pitches be at least 5% out of tune in order to be replaced, but requiring that pitches be more than 20% out of tune limited the benefit of the error-reduction system too greatly to be useful.

For future work, a more consistent key-finding algorithm would be most beneficial to the system. When comparing selections that are in the same key, a good key-finding algorithm should return the same key most of the time. In a query-by-humming system, however, the errors produced by the query process are enough to make the algorithm less effective. A more error-tolerant version of the Spiral Array CEG algorithm would benefit the system.

5. ACKNOWLEDGMENTS

Thanks to Elaine Chew, David Etlinger, Sharol Gersic, and Bryan Pardo.

6. REFERENCES

[1] Birmingham, William et al. The MusArt Music-Retrieval System. D-Lib Magazine, (February 2002).

[2] Boersma, Paul. Accurate Short-Term Analysis of the Fundamental Frequency and the Harmonics-to-Noise Ratio of a Sampled Sound. Institute of Phonetic Sciences (1993).

[3] Chew, Elaine. Modeling Tonality: Applications to Music Cognition. Proceedings of the 23rd Annual Meeting of the Cognitive Science Society. Mahwah, NJ/London. Lawrence Erlbaum Assoc. (2001).

[4] Durbin, Richard et al. Biological Sequence Analysis. Probabalistic Models of Proteins and Nucleic Acids. Cambridge University Press (1999). p. 12-27.

[5] Eerola, T. & Toiviainen, P. (2004). MIDI Toolbox: MATLAB Tools for Music Research. University of Jyväskylä: Kopijyvä, Jyväskylä, Finland. Available at http://www.jyu.fi/musica/miditoolbox/.

[6] Etlinger, David. Matlab Implementation of the Spiral Key Algorithm. (2005).

[7] Gersic, Thomas . Music Melody Matching Machine. Available at http://www.gersic.com/pdf/melody_matching.pdf (2005).

[8] Krumhansl, Carol L. *Cognitive Foundations of Musical Pitch*. New York, NY. Oxford University Press (1990).

[9] Krumhansl, C. L., and Kessler, E. J. Tracing the dynamic changes in perceived tonal organization in a spatial representation of musical keys. Psychological Review, 89, (1982). p. 334-368.

[10] Uitdenbogerd, Alexandra and Zobel, Justin. Matching Techniques for Large Music Databases. Available at http://www.cs.northwestern.edu/~pardo/machine_perception_of_music/uitdenbogerd-acm-multimed-99.pdf