DIRECT ESTIMATION OF MUSICAL PITCH CONTOUR FROM AUDIO DATA

Adriane Swalm Durey and Mark A. Clements

Center for Signal and Image Processing
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332

ABSTRACT

The increasing amount of music being stored in digital formats calls for increasingly more creative methods for music information retrieval. One subset of music retrieval methods relies on storing a melody in a pitch contour representation. Most often, this contour information is generated either from symbolic format (MIDI) or from raw audio after a pitch transcription step. In this paper, we propose a method of extracting pitch contour information from musical audio without an intermediate transcription step by combining a musically-tuned constant Q transform with cross-correlation. When tested on a database of 520 monophonic music recordings, our method generates pitch contours from raw audio data with up to 98% accuracy.

1. INTRODUCTION

As an increasing amount of information, including musical recordings, is made available in digital formats, it becomes increasingly important to access that information in a timely and sensible fashion. An analogy to searching text-based collections illustrates the problems inherent in searching a musical database. If a user is interested in "Beethoven's Fifth Symphony," he can generate a World Wide Web search for that text string and locate information likely to interest him, that is, a group of documents containing those keywords. Now suppose the user wishes to search a library of audio files for the same piece. He is more likely to recall the characteristic "Da-da-da-dum" of the opening measures, shown in Figure 1, than the title or composer of the piece. However, there is no analogous way to access this information directly from most audio file formats. Some new mechanism is required in order to make such an obviously mnemonic description of the contents of music files accessible by the user; this is the heart of the query-by-melody problem which our estimator addresses.

The motivation to estimate pitch contour from raw audio derives from the need for content-based access to archives of digitized music. One of the greatest problems in searching such a database by melodic content is that there are many equivalent, or invariant, methods of representing a given melody as far as a listener is concerned, such as in different keys. Thus, it is desirable for a representation of melody to ignore such invariances when comparing two melodies. A second problem with a music database accessed by content is that users will provide the system with a query which contains some degree of error, for example, by singing a query badly.

One of the primary ways in which these melody matching problems are addressed by query-by-melody systems is to convert the query and database melodies to a pitch contour representation [1]. The pitch interval contour is found by simply calculating the number of pitch steps from one note to the next; for the Beethoven example in Figure 1, this contour would be "X 0 0 -4 +2 0 0 -3." This has the advantage of making all key transcriptions of the same melody equivalent and of reducing the effect of user errors in pitch production. User errors can be further minimized by the use of generalized contours which fold the exact pitch steps into broader categories, such as {Up, Down, Same}, illustrated in this example as, "X S S D U S S D." Unfortunately, the degree of possible melody confusion also increases with coarser contour representations.

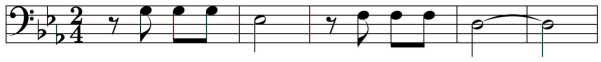

Fig. 1. The opening theme of Beethoven's *5th Symphony*

Much of the seminal work in melody-level music information retrieval (MIR) utilizes pitch contours during processing. For example, McNab, *et al.*, [1] extract contour representations of melody information both from queries to the system and the contents of their musical database. Using dynamic programming, they compare pairs of contours to determine which melodies are most likely to match. However, one common problem with such a system is that employing a pitch contour representation requires either that the contents of the database be stored in a structured audio format such as MIDI or that the music first be transcribed from a raw audio to MIDI before pitch contour is computed. The ubiquitous nature of music in the MPEG Layer 3 encoding on the World Wide Web today illustrates the need to address finding melodies in audio files that that contain no strictly musical information. Additionally, the conversion of music from a format like MPEG Layer 3 to one like MIDI is not a solved problem. Though research on tracking pitch in polyphonic music recordings is ongoing [2, 3], at present no method is completely successful.

The goal of this paper is to describe a method for estimating pitch contour directly from raw audio without requiring that additional musical information be attached to the audio or requiring a definitive pitch transcription step. Such a calculation could be used as a pre-processing step for a melody-level MIR system operating on raw audio data [4]. Section 2 will describe our proposed method for the estimation of pitch contour from raw audio. Evaluation of our system on a test database of raw audio is discussed in Section 3. Results and conclusions are presented in Sections 4 and 5, respectively.

2. ESTIMATION OF PITCH CONTOUR

The estimation of pitch interval contour from raw audio data is a multi-step process designed to overcome two problems associated with musical data. The first problem is that musical pitch in the Western tonal scale is logarithmic. [1] The second difficulty is that we are interested not in finding that we have moved from one specific pitch to another (for example, from C_4 to G_4), but that we have moved over a given pitch interval (in this case, $+5$ pitch steps.) We desire to do so without requiring an additional transcription step (*e.g.*, identifying C_4 and G_4) before making the pitch interval determination to avoid introducing error into the system via that intermediate step.

In order to solve the problem of the logarithmic nature of pitch, we first apply the constant Q transform described by Brown [5]. To approach the problem of determining pitch step from this spectral data rather than from the results of transcription, we utilize cross-correlation, as described in Section 2.2. The generalization of exact pitch interval contour to other contour classes is discussed in Section 2.3.

2.1. Constant Q Transform

The first step in the estimation of pitch interval contour from raw audio is to produce a windowed musical spectrum of that data. This is done using a musically tuned constant Q transform (CQT) defined by Brown [5] as:

$$X[k] = \frac{1}{N_k} \sum_{n=0}^{N_k-1} W_k[n]x[n]e^{\frac{-j2\pi Qn}{N_k}}$$

where $Q = 1/(1-q) \approx 34$ has been selected to provide the appropriate scaling for music. ($q = \sqrt[24]{2}$ is a quarter-tone.) The window width, N_k, is selected to set the center frequencies of the CQT filters to the pitches of the musical scale. In this case,

$$N_k = \left\lfloor \frac{F_s Q}{q^k f_{\text{min}}} \right\rfloor$$

where f_{min} is the minimal center frequency that we are interested in computing, in this case, $f_{\text{min}} = C_2 \approx 65.41$ Hz. We can then define $W_k[n]$ as the window to use for each value of k. In our implementation, $W_k[n]$ is a Hamming window of length N_k.

Calculating the CQT at each time frame provides us with a musical spectrum of the signal, in which every other bin corresponds to one note of the musical scale. An example of this spectrum for the song *Three Blind Mice* is shown in Figure 2. Now, we have a collection of bins which are equally spaced in a musical sense. At this point, we could, if we desired, estimate the note which is sounding at each instance in time, for example, by selecting the maximally valued frequency bin. However, we choose to process the data further before firmly defining the results of the estimator, using the relationship between the bins to directly estimate pitch contour.

2.2. Cross-Correlation

The second step in the estimation of pitch interval contour from raw audio is to calculate the cross-correlation of the absolute value of the CQT between the current and previous frames. First, we drop all the values of the CQT that fall between two notes in the scale, leaving us with only the bins that represent the frequency of some note on the scale. Next, we calculate the cross-correlation of these reduced CQTs. This can be expressed by the following equation:

$$\begin{aligned}(f \star g)_n &= \sum_{k=0}^{T-1} \overline{f[k]} g[n+k] \\ &= \sum_{k=0}^{T-1} |X_{\tau-1}[k]||X_\tau[n+k]|\end{aligned}$$

where $X_\tau[k]$ denotes the CQT of x at time τ.

The nature of a musical audio signal under the CQT is to produce a series of pulses (over background noise) at the fundamental and harmonic frequencies of the sounding note (as shown in Figure 2.) The result of correlating these pulse trains is to determine at what offset the best alignment between frames occurs. The value of the cross-correlation, $(f \star g)$, at n can be taken as the relative likelihood that n is the pitch change for that frame; it can be shown that this value is maximized when n is equal to the pitch interval between two notes. At this stage, we can use the cross-correlation output as a feature for MIR without providing a definitive estimate of the pitch contour; for example, we can take the value of the cross-correlation at each value of $n \in [-11, 11]$ (representing any change up to an octave.) We can also estimate the most likely pitch change for each frame by finding the value of n for which $(f \star g)$ is maximized. Thus, if the maximum of the cross-correlation occurs when $n = -5$, then the estimated pitch interval at that frame is -5. This is shown at the bottom of Figure 2.

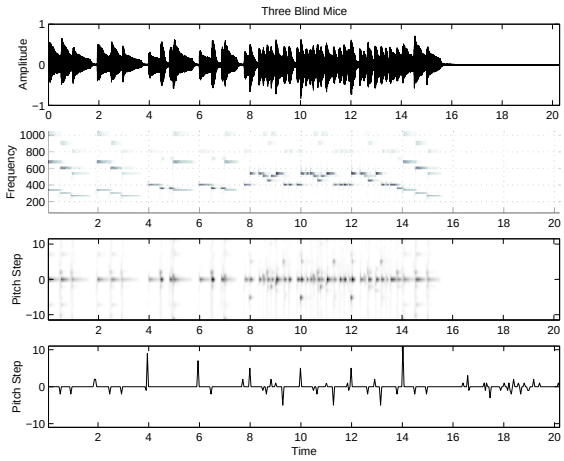

Fig. 2. From top to bottom: Audio waveform, constant Q transform, pitch interval contour, maximal pitch interval contour

2.3. Generalized Contour

While pitch interval contour exactly describes the path a melody takes during the course of a song, that accuracy may harbor errors in the melody when it is used as a query to a musical database. Therefore, we wish to be able to define other generalized classes of melodic contour. Two common examples of such classes are {Up, Down, Same} (C_3 contour) and {Large step up, Large step down, Small step up, Small step down, Same} (C_5 contour) [6]. Again, we will create this estimate without a definitive transcription step or defining the pitch interval contour of the melody prior to estimating the generalized contour.

[1] In the Western tonal scale (hereafter, the musical scale), moving up an octave doubles the pitch of a note. The series of twelve pitches within an octave is divided into 12 equal logarithmic steps (semi-tones).

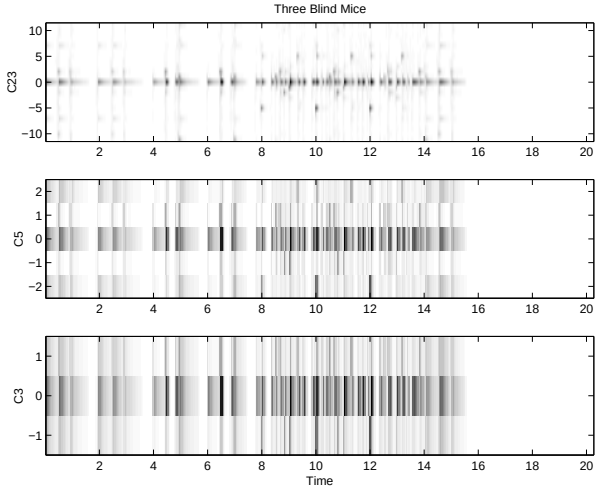

Fig. 3. Examples of generalized contour estimation

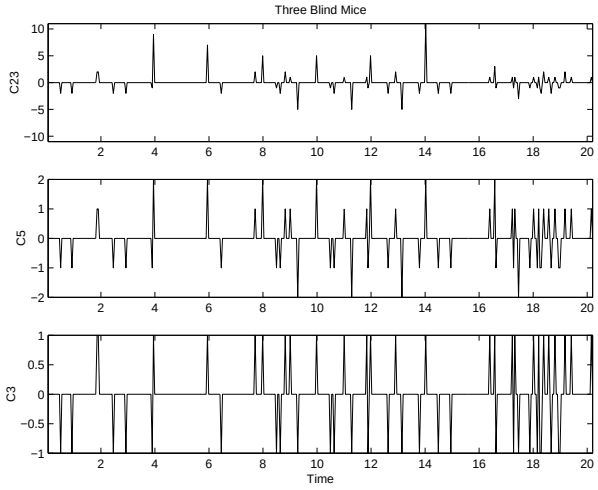

Fig. 4. Examples of maximal generalized contour estimation

Each of the methods which we have tested for the production of generalized contours begins with the output of the cross-correlation step described in the previous section. In the following paragraph, we will assume that we are estimating C_3 contour. First, the values of the cross-correlation at each time frame are grouped into logical sets based on the type of generalized contour to be calculated. For this example, all cross-correlation values are grouped according to $\{n < 0, n = 0, n > 0\}$. Next, several methods were investigated for the combination of the values in those sets into a final value for the generalized contour at that time frame. The tested methods included taking the mean, product, sum, and maximum of the values in each set. In empirical testing on several songs from the evaluation database, it was found that the maximum produced the best results for estimating generalized contour. Several examples of generalized contour and maximal contour estimates for different classes are given in Figures 3 and 4 where C_3 and C_5 contour are defined as above, and C_{23} contour is defined as exact interval contour within $[-11, 11]$ pitch steps.

3. EVALUATION

We evaluate the performance of our pitch contour estimator by comparing the maximal contour estimate with a ground truth contour extracted from our test database. The system is evaluated on a database of monophonic music recordings developed by the authors for melody-based MIR [4]; this is described in Section 3.1. The method of evaluation is then detailed in Section 3.2. The results of this evaluation are discussed in Section 4.

3.1. Database

The database on which this contour estimator is tested consists of 520 recordings created for the purpose of testing a query-by-melody database system. These recordings are of monophonic melodies drawn from a collection of children's songs, such as *Three Blind Mice*, and Christmas carols, such as *Jingle Bells*. A smaller subset of 70 songs was played several times by either an amateur or a trained pianist. Each rendition of the melody was recorded several times in different instrumental voices.

The audio data was collected from a Yamaha keyboard. It was captured by computer at a sampling rate of 22050 Hz using mono audio input and stored as WAV files. This introduced additional complexity to the data set, since it added varying amounts of noise to each recording. At the same time, the corresponding MIDI data was collected from the keyboard memory to act as label data for the music. Since we have identified a transcription step from audio to MIDI as a potential problem in acquiring melodic contour, we must note that the MIDI data is only used in the evaluation of the estimator, not during its calculation.

3.2. Method

Our pitch contour estimator was evaluated by comparing maximal contour estimates from the raw audio data to ground truth contours extracted from the corresponding MIDI data. First, we will describe the creation of the ground truth vector, then the comparison between it and the contour estimate.

Each audio file in the database has an associated MIDI file which was created at the time of its recording. The pitches which occur in a song, p_i, and their onset and offset times, t_i^{on} and t_i^{off}, can all be easily extracted from the MIDI data. ($i \in [1, N]$ and N is the number of notes in the melody.) Then, we can simply subtract the MIDI number of the previous note, p_{i-1}, from each note in the melody, p_i. This provides us with the C_{23} contour of the melody line, c_i^{23}, for $i \in [2, N]$, and a time stamp for each note change (the start time of the next note, t_i^{on}.) The equivalent C_3 and C_5 contours can then be computed by classifying according to the value of the C_{23} contour.

The remaining problem is that the maximal contour estimates are calculated once every sample period, but note onset data occurs only once per note in a MIDI representation. To accommodate this, we will expand the series of pitch contours and time stamps into a ground truth vector. We use the known sample period, T, of the contour estimator to find the sample number for a given note onset time. The ground truth vector is defined (for C_{23} contour) as:

$$g[n] = \begin{cases} c_i^{23}, & n = |t_i^{\text{on}}/T|_R, \text{for } i \in [2, N] \\ 0, & \text{otherwise} \end{cases}$$

where $|x|_R$ stands for rounding. The equivalent vector for C_3 contour can be defined by replacing c_i^{23} with $\text{sgn}(c_i^{23})$ and so on. Examples of the ground truth vectors for *Three Blind Mice* are shown

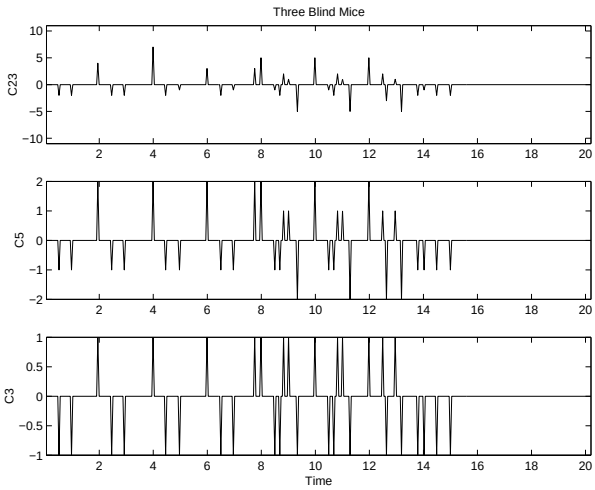

Fig. 5. Examples of contour ground truth vectors

Contour	*Distance*	*Mean*	*Std*	*Min*	*Max*
C_3	Edit	7.05%	2.90%	0.59%	16.71%
C_3	Edit ± 1	1.92%	2.30%	0.00%	12.09%
C_3	Euclidean	6.05	1.47	1.41	10.25
C_{23}	Edit	7.11%	2.91%	0.59%	16.71%
C_{23}	Edit ± 1	2.04%	2.34%	0.00%	12.36%
C_{23}	Euclidean	25.23	8.20	6.93	54.09

Table 1. Error rates for maximal contour estimators

in Figure 5. We now evaluate our method by applying edit and Euclidean distance measures our maximal contour estimates and the corresponding ground truth vectors.

4. RESULTS

In this section, we examine the results of comparing our maximal contour estimates and the ground truth vectors for each song in our database for both C_3 and C_{23} contour. A summary of the results can be seen in Table 1. In the table, edit distance determines whether the contour values at each time frame match and edit distance ± 1 determines if a matching value can be found within one additional time sample in either direction. Edit distances have been normalized by the length of each song. Euclidean distance is defined in the usual way.

The edit distance results show good performance for our contour estimator. For strict edit distance, the contour estimator shows an error rate of $7 \pm 2.9\%$ for both types of contour. When the edit distance ± 1 is used, the mean error drops further to about $2 \pm 2.3\%$ for both contour types. The difference in the mean values is likely to be found in registration errors created by resampling the audio signal versus converting time directly from MIDI data to the sample domain. The reason for the similarity in the standard deviations will be addressed shortly. A comparison of the Euclidean distance measures shows the ability of the C_3 contour to subsume errors in melody comparisons which may be more apparent in finely grained representations like the C_{23} contour.

Two primary types of errors were observed during the evaluation process. One problem is that when there is silence in the piece of music (as for the period from 16–20 seconds in *Three Blind Mice*), many spurious contour estimates are generated. Since there is no clear musical information here, the calculation of the contour of the noise spectrum generates many incorrect values. This implies that there are many errors which cannot be corrected simply by widening our temporal field of view, as illustrated by the two edit distance measures above. These errors should be reducible by pairing the estimator with a music/silence detector. A second obvious problem occurs when a note is not immediately preceded by another note (as for the first note in a piece or for notes occurring after long pauses); there is an ambiguity as to what the pitch interval might be. Again, music/silence detection might allow us to reduce errors of this nature.

5. CONCLUSIONS

In this paper, we have presented a new method for extracting pitch contour information directly from musical audio. We have shown that this method is capable of successfully extracting pitch contour information from monophonic audio in most cases, and when paired with a music/silence detector could be even more successful. It would be simple to apply this estimator to a piece of monophonic audio and use the result as a precursor to dynamic programming in a music retrieval system such as is described in Section 1. It would also be valuable to test this system upon polyphonic audio, to determine if it can provide useful melodic information in that context (as opposed to this monophonic context, where the desired melody is clearly shown in the signal.)

Our primary research focus is to examine further uses of this method outside of a symbolic music representation context. If it proves capable of representing information about the multiple voices of a piece of polyphonic music at the same time, it could be a valuable tool for melody-level MIR. We are currently testing this melody representation (operating from the contour estimate stage in Figure 3 rather than the maximal contour estimate in Figure 4) in conjunction with the hidden Markov model-based melody spotting system described in Durey and Clements [4].

6. REFERENCES

[1] R. J. McNab, L. A. Smith, D. Bainbridge, and I.. H. Witten, "The New Zealand Digital Library MELody inDEX," *D-Lib Magazine*, May 1997.

[2] M. Goto, "A predominant-F0 extimation method for CD recordings: MAP estimation using EM algorithm for adaptive tone models," in *Proc. ICASSP*, Salt Lake City, UT, May 2001, Electronic Proceedings.

[3] T. Virtanen and A. Klapuri, "Separation of harmonic sounds using linear models for the overtone series," in *Proc. ICASSP*, Orlando, FL, May 2002, vol. II, pp. 1757–1760.

[4] A. S. Durey and M. A. Clements, "Features for melody spotting using hidden Markov models," in *Proc. ICASSP*, Orlando, FL, May 2002, vol. II, pp. 1765–1768.

[5] J. C. Brown, "Calculation of a constant *Q* spectral transform," *J. Acoust. Soc. Amer.*, vol. 89, no. 1, pp. 425–434, January 1991.

[6] J. S. Downie, *Evaluating a Simple Approach to Music Information Retrieval: Conceiving Melodic N-Grams as Text*, Ph.D. dissertation, Univ. Western Ontario, 1999.