Proceeding of 3rd International Symphosium on Image and Signal Processing and Analysis (ISPA03)
Italy, Rome, Sep. 18-20, 2003

A Robust Singing Melody Tracker Using Adaptive Round Semitones (ARS)

Chong-kai Wang
Dept. of Electrical Engineering, Chang Gung University, Taoyuan 333, Taiwan
ckwang.ee86@nctu.edu.tw

Ren-yuan Lyu
Dept. of Electrical Engineering, Chang Gung University, Taoyuan 333, Taiwan
rylyu@mail.cgu.edu.tw

Yuang-chin Chiang
Institute of Statistics, National Tsing Hua University, Hsinchu, 300, Taiwan
ghiang@stat.nthu.edu.tw

Abstract

In this paper, an approach for melody tracking is proposed and applied to applications of automatic singing transcription. The melody tracker is based on Adaptive Round Semitones (ARS) algorithm, which converts a pitch contour of singing voice to a sequence of music notes. The pitch of singing voice is usually much more unstable than that of musical instruments. A poor-skilled singer may generate voice with even worse pitch correctness. ARS deals with these issues by using a statistic model, which predicts singers' tune scale of the current note dynamically. Compared with the other approaches, ARS achieves the lowest error rate for poor singers and seems much more insensitive to the diversity of singers' singing skills. Furthermore, by adding on the transcription process a heuristic music grammar constraints based on music theory, the error rate can be reduced 20.5%, which beats all the other approaches mentioned in the other literatures.

1. Introduction

Traditionally, music transcription relied on well trained music professionals, who were usually good at playing one or more musical instruments and well acquainted with music theory. They usually composed the melodies by musical instruments and transcribed it into musical notation. A person who is not provided with professional training about music will find it very difficult to compose music even thought he has a very good melody in his mind. One of the ways to keep such kinds of good melodies is that he could sing to music professionals such that the professionals could transcribe the melodies by listening to the singing and writing down the notes. If there exists an intelligent software which could transcribe humans' singing into music notes automatically, more people will be fascinated with music composing and more splendid melodies will be created all over the world. Some computer-aided composing systems have indeed been on the markets. However, according to the authors' knowledge, very few of them provide the mature functions for automatic transcription.

Singing transcription also plays an important role in query-by-humming systems, which allow users to access the music database by "singing" a melody as a query.[1] Such a kind of content-based information retrieval technology is a revolution on multimedia data processing. When a person wants to find a music piece from a music database, he will no longer worry about not remembering the name of the author or the title of the song. He can only hum to a microphone and then get whatever music he wants to listen to. After all, a person can remember the music melody of a song much easier than the title of the song or the name of the author. Singing transcription system makes music more convenient to be created and more accessible. Any one could be a composer if there is a good melody flowing through his mind. A robust singing transcription system could help not only the music professionals but also normal people who feel interested in music but lack the professional training.

In this paper, we describe a singing transcription system, which could be divided into two modules. One is for the front-end voicing processing, including voice acquisition, end-point detection and pitch tracking, which deal with the raw singing signal and convert it to a pitch contour. The other is for the melody tracking, which maps the relatively variation pitch level of human singing into accurate music notes, represented as MIDI note number. The overall system block diagram can be shown as Figure1.

In front-end voice processing level, we have designed a special user interface suited for singing input derives from our previous researches on automatic continuous speech recognition system.

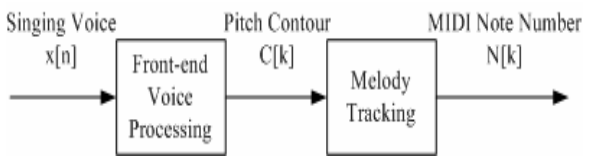

Figure 1. The Singing Transcription System

Although there were many state-of-the-art algorithms for the front-end voice processing, few algorithms could be found mature enough for the melody tracking in the literatures. In the melody tracking module of our system, we not only implement several algorithms found in the literature, but also propose a new melody tracking algorithm called Adaptive Round Semitones (ARS) to improve the performance of the system for poor singers. Furthermore, by adding on the transcription process a heuristic music grammar constraints based on music theory, the error rate can be reduced to the lowest among the other proposed approaches in the previous literatures.

This paper is therefore organized as follows: Section 1 is this introduction. Section 2 is a brief description for what we have used in the front-end voice processing module. Section 3 is an overview for the melody tracking algorithms proposed in the literatures and implemented in this paper for comparison. Section 4 is for the newly proposed ARS algorithm in detail, plus a brief description about adding some music grammar constraints to reduce the transcription error rates further. Section 5 is for the experiments to compare the performance of the different algorithms implemented in this paper. Finally, section 6 is a conclusion.

2. Front-end voice processing

2.1. End-point detection

There are many state-of-the-art techniques for the front-end voice processing. First of all, the voice signal should be end-point detected to determine the starting and ending point of one utterance of singing. A general end-point detection algorithm does not work well without involving some dynamic programming based algorithms, like Hidden Markov Model (HMM) based speech detection. To simplify the design and focus our effort in the melody tracking module, we put constraints on the singer to hum a melody by singing the syllables with a stop consonant as the initial part of that syllable, such as “ba”, “da”, ”di”…etc. Thus, the end-point of each syllable could be simply determined by energy of the utterance.

2.2. Pitch tracking

Another even more important issue in the front-end voice processing module is to extract pitch or fundamental frequency (F_0) of human singing signal and segment the pitch contour into several pitch sections, each of which will correspond to a musical note. There are many algorithms to extract pitch from audio signal, primarily developed for speech processing related researches. Here, we adopt autocorrelation function method for simplicity. [2]

The extracted pitch contour of a singing signal is usually neither continuous nor smooth enough. Thus some heuristic smoothing techniques should be applied.

2.3. Island Building

The pitch value in the vowel region of a syllable is usually well defined and approximately a constant. However, the pitch value of the consonant region and silence region would be random and should be filtered out. The process of filtering out the pitch contour in the consonant and silence part of an utterance is called Island Building. [3] An instance of Island Building was shown as in Figure 2. , and each “Island” is referred to a note in music score.

Notice that the vertical axis of Figure 2. is the singing pitch in unit of *cent*, which will be described in the follow section. The sum average of the pitch values within each “Island” was then calculated and waited to be processed in the melody tracking module.

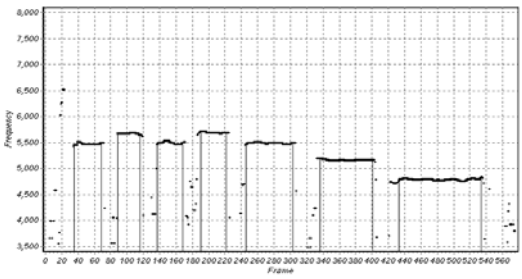

Figure 2. An instance of Island Building for pitch contour

3. Melody Tracking

The melody tracking module of the system deals with the average pitch value f (frequency in Hz) for each pitch island described in the previous section and will transcribe them into musical notes, which have been defined as MIDI numbers for all possible semitones in the computer music processing domain.

The pitch value *f* and the corresponding MIDI number ***N*** can be related as the following formula:

$$N = 69 + 12 \times \log_2 \frac{f}{440}$$

For examples, the note "concert A" with pitch being 440 *Hz* will be assigned a MIDI number 69, and the note A#, which is just one semitone higher than "concert A", is then assigned 70 as its MIDI number. On the equal tempered scale, the pitch interval between two consecutive MIDI numbers could be divided into 100 *cents*, e.g., the note "concert A" is 6900 cents.

Since the pitch interval between two MIDI numbers is divided into 100 cents, it is straightforward to convert pitch in cents to MIDI number by rounding the real value of $\text{pitch in cents}/100$ to the nearest integer. Such an approach, called *Round MIDI* as mentioned in the literature, is analogous to the uniform quantization used in speech coding algorithm, such as linear PCM, where the value of each speech sample is uniformly quantized with equal step size and assigned a quantization level. However, human's pitch generation process is much more unstable than that of musical instruments. An untrained singer may generate voice with even worse pitch correctness. For example, the pitch level will drift upward or downward due to the singer's mood, and the pitch interval may not keep constant during singing. Just like a successful speaker independent speech recognition system should do, an intelligent automatic music transcription system should tolerate the incorrectness of pitch generated from a variety of naïve users. The experiments done in the other literatures show that the straightforward Round MIDI algorithm did not work well for melody tracking, thus several improvements have been proposed.

One improvement was proposed by McNab et al. [3] Based on the assumption that the user is singing to the equal tempered scale, but then adjusting the scale during transcription, they introduced a procedure to adjust the scale by using a constantly changing offset, initially estimated by the deviation of sung tone from the nearest tone on the equal tempered scale. The resulting musical scale is continuously altering the reference tuning, in relation to the previous note. This approach was implemented and referred to as *McNab's Moving Tuning* in this paper.

Another improvement was proposed by Haus et al. [4] They assumed every singer had his/her own reference tone in mind and he/she sang each note relatively to scale constructed on the tone. They also presume that error do not propagate during singing and are constant, apart some small increases with size of the interval. Thus, they estimated a reference deviation, adjusted the note pitch, and then adopted the basic Round MIDI algorithm mentioned above. This approach was also implemented and referred to as *"Haus' Modified Round MIDI"* here in this paper.

Round MIDI is an absolutely approximating technique directly applied on the MIDI number, or equivalently on the music notes; *McNab's Moving Tuning* first adjusts a constantly changing offset to each music note, and then does the following as Round MIDI; *Haus' Modified Round MIDI* estimates an offset from all notes, adjusts all notes with this offset, and then does the following as Round MIDI, too. However, it was said that only about 1 in 10000 people claim to have tone-absolute pitch. [5] It is more sensible to apply the rounding techniques on the pitch intervals or differences instead of the pitch values. This idea leads to another types of algorithms based on tone-relative pitch notion, such as *"Round Intervals"* algorithm. Round Intervals is as literal as it is named. It evaluates the pitch difference of neighboring two notes and then rounds it to the nearest tone offset. Although "Round Intervals" achieves lower error rate than the other types of "Round MIDI" algorithm, it does not adapt to singers' variation tune scales and pitch differences. Thus, although it performs well for good-skilled singers, the error rate will significantly increase for poor-skilled singers. To address this issue, we proposed a new technique, called *Adaptive Round Semitones (ARS)*, by using an adaptive autoregressive model to dynamically changing the tuning scale. This approach will be described in details in the following section.

4. Adaptive Round Semitones (ARS)

The ARS was designed on the following three assumptions:

1. Most singers have only tone-relative pitch.
2. The tune scale of human singing voice is not necessarily 100 cents per semitone.
3. The tune scale will probably change with time while singing and be dependent on previous notes.

For example, while a singer singing a rising melody sequence, he will inadvertently sing higher and higher. For a poor singer, the melody sung outward does not always match what he wishes to sing, although the listeners usually can understand what he sang. This is because he has a changeable and unusual tune scale. One important motive of designing ARS is to deal with these issues.

4.1. Statistic model

The block diagram of ARS is shown in Figure3. , where the input is the sequence of the pitch in *cents*, C[k], and the output is the sequence of the MIDI numbers, or equivalently the music notes, N[k].

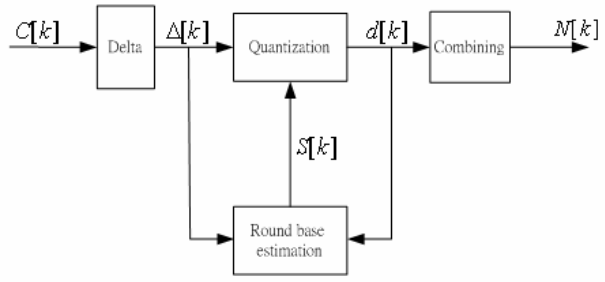

Figure 3. Adaptive Round Semitone (ARS)

All the state variables shown in Figure 3. will be described as follows:

$$\Delta[k] = C[k] - C[k-1]$$

$$d[k] = \begin{cases} R\left(\dfrac{\Delta[k]}{100}\right) & 1 \le k \le T \\ R\left(\dfrac{\Delta[k]}{S[k]}\right) & T+1 \le k \le K \end{cases}$$

$$S[k] = \begin{cases} \dfrac{\sum_{m=1}^{T} a[m] \cdot \Delta[k-m]}{\sum_{n=1}^{T} b[n] \cdot d[k-n]} & T+1 \le k \le K \\ 100 & 1 \le k \le T \end{cases}$$

$$N[k] = N[k-1] + d[k]$$

where Δ[k] is the input pitch difference (in cents), d[k] is the difference of the output MIDI numbers, N[k] is the final output MIDI number sequences, and $R(x)$ is a rounding function which rounds off the real number x to the nearest integer, and S[k] is the estimated tuning scale, which is initially set to 100 according to the assumption of equally tempered scale and is dynamically adjusted according to the information provided by the previous music notes.

Here we assume an autoregressive moving average (ARMA) model with parameter set {T ; a[1], a[2],…a[T]; b[1], b[2],…b[T]} to estimate S[k]. The parameter set was determined empirically in this paper.

4.2. Rule-based post-processing

Furthermore, a lower and upper bounds for the tune scale S[k] is set to avoid over-tuning. A center-clipping and side-clipping process will be done to compensate for the overly tuning scale.

Besides, some local rules are added to deal with some exceptional cases:

1. Initial effect of tone: if there are continuous notes identified to the same MIDI number, the tune scale S[k] will be set back to 100 as the initial value.
2. Memory effect of tone: if pitch of the present note is near the second previous note, they will be identified to the same MIDI number.

4.3. Music grammar constrains

Most songs are composed by conforming to an underline music grammar. Tone distribution of a song melody is not fully randomized. A well composed song could be clustered in music theory by its tunes, such as C-Major, E-minor…etc. A Major scale is a structure of tones, namely “Do”, “Re”, “Mi”, “Fa”, “So”, “La”, “Si”, “Do”, where the differences in semitones of each continuing tones are 2,2,1,2,2,2,1. Such a tone structure could be looked upon as a grammar to put a constraint on the possibility of notes, which can be followed by another note. For example, if the pitch differences in semitones of three notes N_1, N_2, N_3, were identified as 1 and 2, then these three notes will be one of the sequences “Mi-Fa-So” and “Si-Do-Re”. If there comes a confusing note N_4 which is 1 or 2 semitones higher than N_3. We can easily determine that N_4 is 2 semitones higher according to the Major scale structure distribution.

Form music theory of Bach’s well tempered scale, a semitone-level shifting of a sequence melody will not change its inner relation. The singer’s singing tune is not unique results from his/her singing key. Equal temperament made possible modern keyboard music with full modulation between all keys. The music grammar constrain is a method that adjust singer’s tune and song’s tune in semitone-level.

The idea that adding constraints of music grammar comes from the language modeling in speech recognition research, which improve the overall recognition rate significantly.

5. Experiments

Experiments have been carried out with different approaches mentioned in the previous sections for comparison with the proposed one. Testing samples were collected from 13 people. Each person sings 9 pieces of melodies. While singing, the singing key and tempo is free. These testing pieces include melodies from several regions and multiple variations:

1. A musical scale (interval between one octave forward and return).
2. Chinese folk song: Jasmine.
3. Bach, minuet in G.
4. Happy Birthday to You.
5. Beethoven, Ode to Joy.
6. Taiwan Ali aborigine song: The green mountain (Gau Shan Cing).
7. Taiwanese folk song: Thinking of You on a Rainy Day.
8. Fishing songs.
9. Edelweiss.

Total number of testing notes is 2885. We consider them as poor singers if the error rate of Round MIDI algorithm exceeds 30%. There are 4 people (869 notes) in the set of poor singers, and 9 people (2016 notes) in the set of normal singers.

The error rates for each implemented algorithms were listed as in Table 1. , where error rates for all singers, normal singers, and poor singers are listed for all 6 melody tracking algorithms implemented or newly proposed in this paper. One can see that ARS plus music grammar not only achieves the lowest error rates 20.2% for all singers but also achieves the lowest performance degradation from the set of normal singers to the set of poor singers, i.e., least insensitive to the variety of multiple singers. With exception to the ARS plus grammar, even though prime ARS does not beat McNab's Moving Tuning algorithm in error rate for all singers, it indeed achieves less performance degradation (5.7%) than the others.

It is also interesting to note that "McNab's Moving Tuning" achieves the same performance as "Round Intervals" from view points of experimental results. By further theoretical analysis, it can also be shown that they are in fact equivalent in algorithm level. By the way, Haus's paper concluded very differently from ours when talking about McNab's Moving Tuning algorithm. It seems that Moving Tuning is not as no good as he said in his paper.

We have developed a WIN32 demo system (shown in Figure 4.) using C++ programming language. The experiment (including data collection, signal processing and error rate verification) is automatically processing by the demo system.

6. Conclusion and future work

Adaptive Round Semitone assumes that the changing of tune scale will not only depend on the singer's singing-skill but is time-variant; and it is simulated by an adaptive ARMA model. ARS achieves the lowest error rate for poor singers and seems much more insensitive to diversity of singers' singing skills. Furthermore, by adding on the transcription process a heuristic music grammar constrains based on music theory, the error rate is can be reduced to 19.8%, which beats all the other approaches mentioned in other literatures.

Owing to the successfully improvements, authors are encouraged to enhance the music grammar model. Tune Map [6] achieves a robust "musical language model" by establishing a tone state probability transition diagram. Viterbi algorithm is applied to estimate the melody tone path. Finally, a speech recognition technique can be applied to improve Island Building process, which segments the voice and unvoiced region accurately. Singers are accepted to sing naturally rather than singing "da" or "ba" instead.

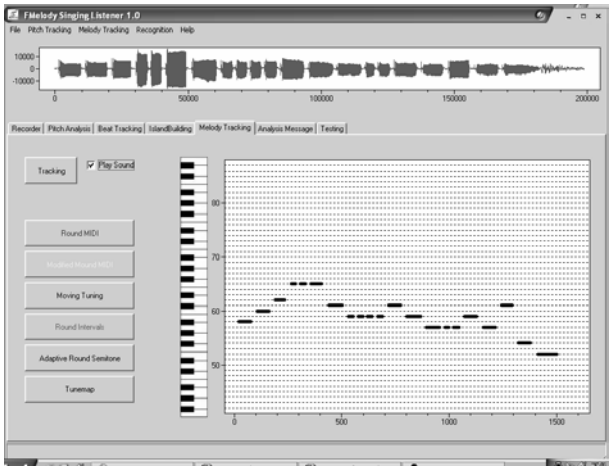

Figure 4. Demo singing transcription system

7. References

[1] A. Ghias, D. Logan, D. Chamberlin, and S.C. Smith, "Query by humming – musical information retrieval in an audio database," *Proc. of ACM Multimedia'95*, San Francisco, Ca., Nov. 1995.

[2] M.M. Sondhi., "New method of pitch extraction," *IEEE Trans. Audio Electroacoust*, AU-16:262-266, 1968.

[3] R.J. McNab, L.A. Smith, and I. Witten, "Signal Processing for Melody Transcription," *Working Paper 95/22, Dept. of Computer Science*, University of Waikato, New Zealand, August 1995.
[4] G. Haus, and E. Pollastri, "An audio front end for query-by-humming systems," *In Proc. of International Symposium on Music Information Retrieval (ISMIR)* ,2001.
[5] Profita, J. and Bidder, T.G.,"Perfect Pitch," *American Journal of Medical Genetics*, 29, 763-771, 1988
[6] Chong-kai Wang, Ren-yuan Lyu, and Yuang-chin Chiang, "Melody Tracking Using Tune Map," (submitted)

Table 1. The experimental result for melody tracking algorithms implemented and newly proposed in this paper

	Round MIDI	McNab's Moving Tuning	Haus' Modified Round MIDI	Round Intervals	**(Prime) Adaptive Round Semitone**	**Adaptive Round Semitone plus music grammar**
Error rate for normal singers	28.1%	20.7%	29.6%	20.7%	**22.8%**	**19.8%**
Error rate for poor singers	33.6%	25.3%	33.1%	25.3%	**24.1%**	**20.5%**
Error rate for all singers	29.8%	22.0%	31.1%	22.0%	**23.1%**	**20.2%**
Relative Performance degradation from normal to poor singers	19.6%	22.2%	11.8%	22.2%	**5.7%**	**3.5%**