Parameter Optimization by a Genetic Algorithm for a Pitch Tracking System

Yoon-Seok Choi and Byung-Ro Moon

School of Computer Science & Engineering, Seoul National University
Sillim-dong, Gwanak-gu, Seoul, 151-742 Korea
{mickey, moon}@soar.snu.ac.kr

Abstract. The emergence of multimedia data in databases requires adequate methods for information retrieval. In a music data retrieval system by humming, the first stage is to extract exact pitch periods from a flow of signals. Due to the complexity of speech signals, it is difficult to make a robust and practical pitch tracking system. We adopt genetic algorithm in optimizing the control parameters for note segmentation and pitch determination. We applied the results to *HumSearch*, a commercialized product, as a pitch tracking engine. Experimental results showed that the proposed engine notably improved the performance of the existing engine in *HumSearch*.

1 Introduction

As information systems advance, databases include multimedia data such as images, musics, and movies. For music databases, there are many search methods based on melodic contours, authors, singers, or lyrics. People want to find music with a few notes that are entered by a convenient method like humming. We aim to develop a pitch tracking engine for a music retrieval system that supports queries by humming.

A number of music search algorithms have been proposed. Handel [1] emphasized that the melodic contour is the most critical factor that listeners use to distinguish a song from the others. In other words, similar melodic contours make listeners consider songs as the same. Music search by humming matches songs and input queries according to properties such as melodic contours and rhythmic variations. Although being attractive, it contains some difficult processes such as melodic contour representation, note segmentation, and pitch period determination. Chou *et al.* [2] proposed a chord decision algorithm which transforms songs and queries into *chord strings*. Ghias *et al.* [3] used *melodic contours* of hummed queries, which consist of a sequence of relative differences in pitch between successive notes. They used an alphabet of three possible relationships between pitches: "U", "D", and "S". Each alphabet represents the situation that the current note is above ("U"), below ("D"), or the same ("S"), respectively, as the previous one.

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 2010–2021, 2003.
© Springer-Verlag Berlin Heidelberg 2003

To represent a melody by a melodic contour, a high-precision pitch tracking process is required. Pitch tracking is a process to determine the pitch period of each note in a melody. It involves two main processes: *note segmentation* and *pitch determination*.

Note segmentation determines where notes begin and terminate, and extracts the notes from a flow of signals. Rodger *et al.* [4] performed note segmentation in two ways: One based on amplitude and the other on pitch. Ahmadi *et al.* [5] presented an improved system for voiced/unvoiced classification from hummed queries based on statistical analysis of cepstral peaks, zero-crossing rates, and energy magnitudes of short-time speech segments. Mingyang *et al.* [6] proposed a system for the pitch tracking of noisy speech with statistical anticipation.

Pitch determination calculates pitch period of a segmented note. There are several approaches to calculate the basic frequency such as autocorrelation [7], AMDF [8], and cepstrum analysis [9]. There are also a number of studies under noisy environments. Shimomuro *et al.* [10] used weighted autocorrelation for pitch extraction of noisy speech. Kunieda *et al.* [11] calculated the fundamental frequency by applying autocorrelation in order to extract periodic features.

Few people usually have perfect pitch and they cannot remember all the exact pitches of their favorite songs either, even when they sing the songs very often. There may be errors as well in the melodic contours extracted from hummed queries. For this reason, we need approximate pattern matching. Our engine is utilized in *HumSearch* which supports queries by humming with an advanced approximate pattern matching.

In this paper, we suggest a genetic algorithm for pitch tracking that transcribes a hummed query into a melodic contour under noisy environments. We not only used the classical methods for pitch tracking such as the energy of short-time speech segment, zero-crossing rate, and cepstrum analysis, but also designed a contour analysis model. While the classical algorithms determined the threshold for note segmentation using a statistical method [5] [6] or an adaptive method [4], we optimize the control parameters of the methods by combining the classical methods with the genetic framework to enhance the performance of the pitch tracking engine.

The remainder of this paper is organized as follows. In Section 2, we summarize preliminaries for pitch tracking. In Section 3, we explain our additional methods such as contour analysis for note segmentation and describe our system in Section 4. We perform experiments and compare the results with an existing pitch tracking engine in Section 5. Finally, we make our conclusions in Section 6.

2 Preliminaries

2.1 Note Segmentation

In pitch tracking systems, note segmentation is the most important process. To transcribe a hummed query into a melodic contour, we need to compute the number of notes contained in the query. Figure 2 shows an energy diagram of

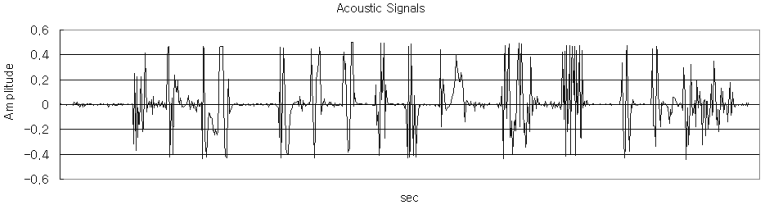

Fig. 1. Original vocal waveform (spaced by 0.01s)

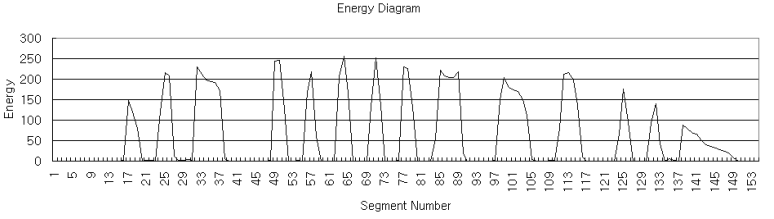

Fig. 2. Energy diagram of short-time speech segment

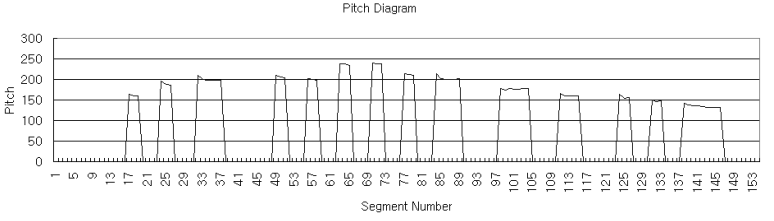

Fig. 3. Pitch diagram of speech segment

a hummed query. Figure 3 is the result of pitch determination from the energy diagram.

Speech Segment. Because of a temporary varying occurrence factor, speech processing is naturally a non-stationary process. However, we assume that the speech signal is short-time stationary for simple speech processing. We divide the whole sound signal into small speech segments and try to determine pitches on these small blocks. It is accomplished by multiplying the signal by a window function, w_n, whose value is zero outside some defined range. The rectangular window is defined as follows:

$$w_n = \begin{cases} 1, \text{ if } 0 \leq n < N \\ 0, \textit{otherwise}. \end{cases}$$

where n is the time point of speech samples and N is the window size. Note that if the size of the window is too large, our assumption becomes unreasonable.

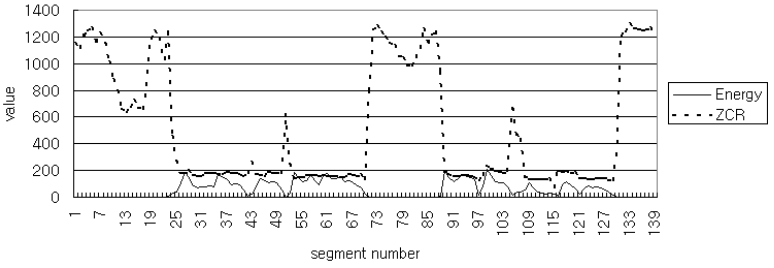

Fig. 4. Note segmentation based on zero-crossing rate

In contrast, if the size of the window is too small, we lack enough information for delimitating a pitch. The size of the window is thus an important factor of determining pitch period.

Energy of Speech Segment. The energy of a speech segment is defined to be the sum of amplitudes of the input wave as follows:

$$E = \sum_{i=0}^{N} |x(i)|$$

where $x(i)$ is the amplitude of original waveform signals and N is the size of a speech segment [12]. A statistical metric such as average or median is used to determine a threshold which classifies segments with or without voice. To make classification simpler, a user sings by "*da*" or "*ta*" because the consonants cause a drop in amplitude at each note boundary [3].

Zero-Crossing Rate. Zero-crossing occurs when neighboring points in a wave have different signs. The zero-crossing rate measures the number of zero-crossings in a given time interval [5]. The zero-crossing rate corresponding to the ith speech segment is computed as follows:

$$ZCR_i = \sum_{n=1}^{N-1} |sgn[x_i(n)] - sgn[x_i(n-1)]|$$

where $x_i(n)$ is the value at position n in the ith speech segment and N denotes the size of the speech segment, $x_i(n)$. A segment with a high zero-crossing rate is judged to be unvoiced or sibilant sounds; a segment with a low number is judged to be voiced intervals (See Figure 4).

2.2 Pitch Determination

Algorithms for determining the pitch on a vocal signal are roughly classified into two types: time-domain methods and frequency-domain methods. The time-domain methods such as autocorrelation and AMDF examine the structure of the sampled waveform, and the frequency-domain methods such as cepstrum analysis perform the Fourier transform and examine the resulting cepstrum.

Autocorrelation. Autocorrelation is one of the oldest classical algorithms for pitch determination. It shows the similarity in phase between two values of the speech signal at times x_n and x_{n+k}. The autocorrelation function $R(k)$ is calculated from a numerical sequence as

$$R(k) = \sum_{n=1}^{N-k} x(n) \cdot x(n+k)$$

where k is the correlation distance in the time sequence, N is the length of the sequence, and $x(n)$ is the value at position n in the time sequence. The function value with respect to k expresses the average correlation between numbers separated by distance k in the sequence. The correlation distance k with the maximum function value corresponds to the pitch period of the input signal.

Cepstrum Analysis. The cepstrum is defined to be the real part of the inverse Fourier transform of the log-power spectrum of $x(n)$ which is the value of acoustic signal in time sequence. Since cepstrum analysis requires a high computation cost, we use the *Fast Fourier Transformation* (FFT). For the FFT, we restrict the window size of the speech segment to be a dyadic number. Figure 7 shows the pitch period extracted from the vocal waveform of Figure 6.

2.3 The MIDI Notes Presentation

Since musical units such as *octaves or cents* are relative measures, we use the MIDI notation for note presentation. MIDI is a standard for communicating with electronic musical instruments and also a standard representation of the western musical scale. It is thus appropriate for representing a *melodic contour* of song or hummed input. MIDI assigns an integer in [0, 127] to each note. Middle C (C4) is assigned 60, the note just above is 61, and that below is 59. MIDI note 0 corresponds to 8.1279 *Hz* and the highest defined note, 127, corresponds to 13,344 *Hz* in frequency. The output of the pitch tracking system is eventually stored in MIDI form. Figure 12 contains a sequence of 12 MIDI notes from a hummed input.

3 Enhanced Methods for Pitch Tracking System

Since a user hums at a slide, the amplitude of an unvoiced sound segment does not drop. Thus the repeated notes may not be segmented successfully using the

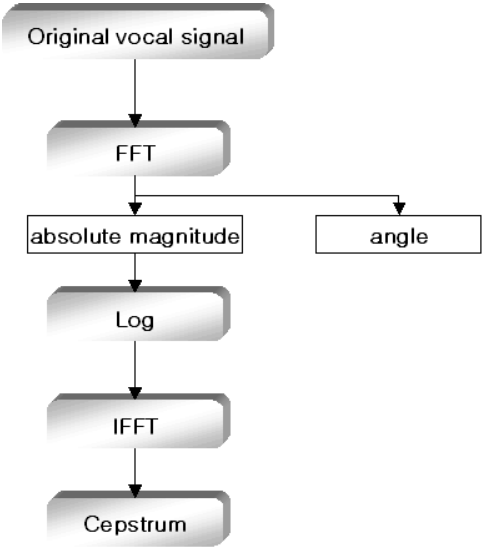

Fig. 5. Cepstrum analysis

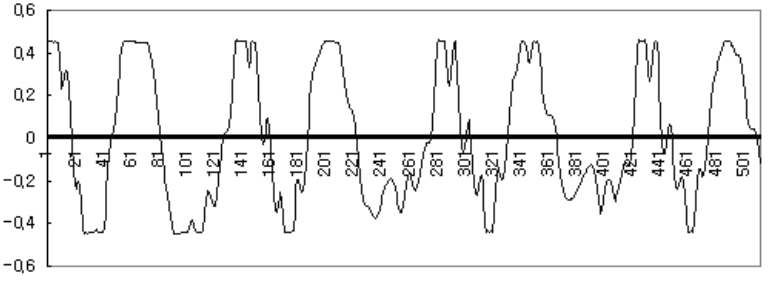

Fig. 6. Original waveform signal

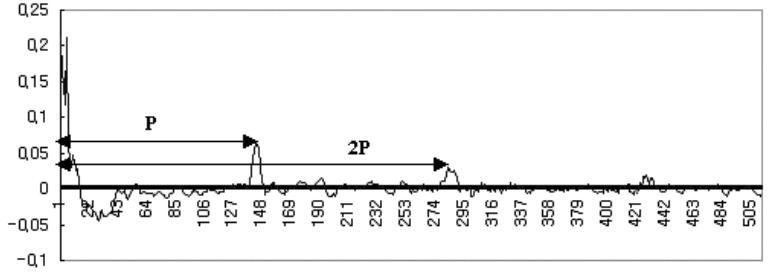

Fig. 7. Cepstrum pitch determination (p = pitch period)

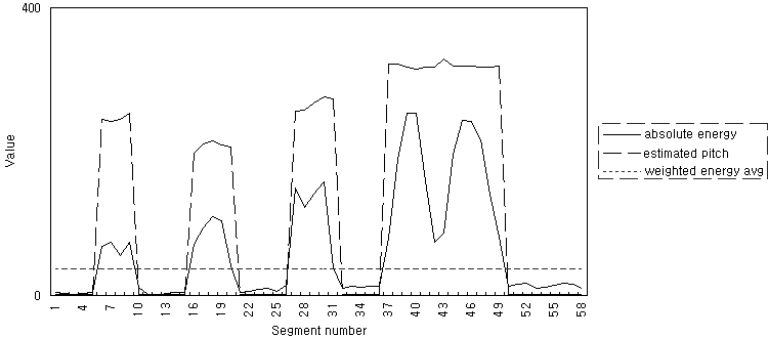

Fig. 8. Note segmentation based on energy of short-time speech segment

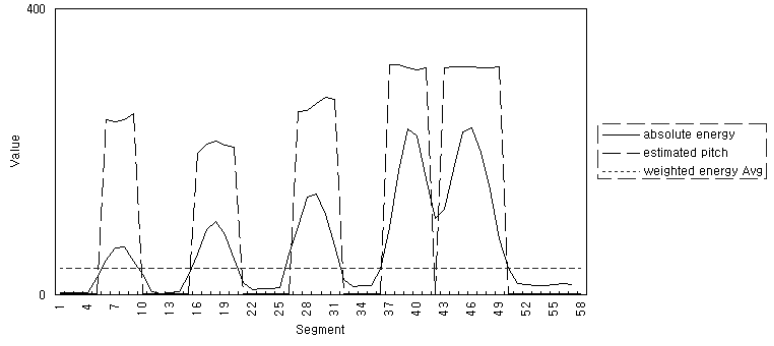

Fig. 9. Note segmentation based on smoothed energy diagram for detecting a valley

methods such as energy of the speech segment and zero-crossing rate. Unsegmented notes should separate into two notes (See Figure 8). We try to separate the successive notes with contour analysis of energy diagram. We detect the valley in the energy diagram and adopt it as a new feature for note segmentation. We use a moving average method and make an energy diagram flatten to remove the small valleys. This method divides the successive notes into two notes (Figure 9).

In our system, we divided the resolution of the relative pitch differences into 15 levels (U1 $\sim$ U7 (Up), D1 $\sim$ D7 (Down), and R (Same)). If the difference is beyond the range, it is assigned one of the boundary levels, U7 or D7. More accurate representation of pitch differences in successive notes is possible by this fine-grained strategy.

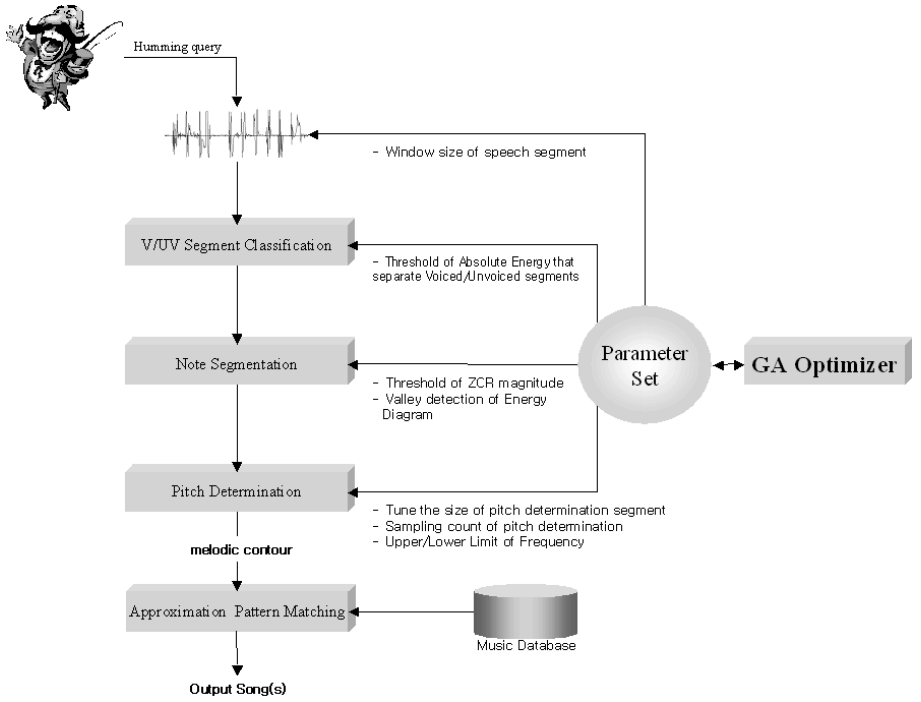

Fig. 10. Optimized pitch tracking system with GA

4 Optimized Pitch Tracking System with GA: GAPTS

4.1 System Architecture

The system architecture is shown in Figure 10. A hummed query fed into the system is evaluated through the pitch tracking and query systems. The efficiency of the pitch tracking module depends on the control parameters such as weighted value, window size, etc. There are three main modules in the system : *voiced/unvoiced classification, note segmentation,* and *pitch determination.*

4.2 GA Framework

Encoding. A chromosome consists of 11 genes. Each gene corresponds to a feature that affects controlling the pitch tracking system. The function of each gene is explained in Table 1.

Initialization. Initial solutions are generated at random. We set the population size to be 100.

```
Create initial population;
Evaluation all chromosomes ;
do
{
    Choose parent1 and parent2 from population ;
    offspring = crossover (parent1, parent2) ;
    mutation(offspring) ;
    evaluation (offspring) ;
    replace(parent1, parent2, offspring);
} until (stopping condition) ;
return the best solution;
```

Fig. 11. Genetic algorithm framework

Table 1. The functions of genes

Genes	Description
0	The window size of speech segment
1	A weighted value for determining the threshold of short-time energy which divides voiced/unvoiced segments
2	The upper bound of frequency considered as voiced pitch
3	The lower bound of frequency considered as voiced pitch
4	Weighted value for determining the threshold of zero-crossing rate which divides voiced/unvoiced segments
5	Boundary value for deciding a sampling count to determine pitch periods
6	The window size of analysis section for pitch determining
7	A window size of the moving average method for smoothing the contour of a short-time energy diagram. If the size is set to 0, the method is not applied to the system
8	The window size of the moving average method for smoothing the contour of a zero-crossing rate diagram. if the size is set to 0, the method is not applied to the system
9	Sampling count for pitch determining
10	Weighted value to determine the threshold that cuts a note into two or more for note segmentation with contour analysis

Parent Selection. We assign to each chromosome a fitness value. A fitness value is defined to be the sum of edit distances between the melodic contours of the hummed query and the original song. Let $SED(t, q)$ be the string edit distance between a hummed query q and the target contour t. The fitness value F_i of chromosome i is defined as follows:

$$F_i = \sum_{i=1}^{N} SED(Q(i), T(i))$$

where $Q(i)$ is the ith hummed query, $T(i)$ is the ith target contour, and N is the number of the hummed queries. We use the tournament selection method [13]. The tournament size is 2.

Crossover. We use the uniform crossover [14]. The relatively high disruptivity of the uniform crossover helped our algorithm escape from local optima and converge to better solutions.

Table 2. sets of parameters

gene number	Optus PTS	GAPTS
0	11	11
1	0.4	0.47
2	530.0	454.91
3	90.0	64.28
4	0.4	0.5
5	6	10
6	2	3
7	0	1
8	0	0
9	1.5	5.00
10	1.5	0.68

Mutation. We randomly select each genes with a probability (= 0.5) and mutate the gene values within its admitted range.

Replacement. After generating an offspring, GAPTS replaces a member of the population by the offspring. We replace the inferior of the two parents with the offspring if the offspring is not worse than both parents. Otherwise, we replace the worst member of the population by the offspring. This scheme is a compromise between preselection [15] and GENITOR-style replacement [16].

Stopping Condition. The GA stops if the generation count reaches a predetermined number, or it shows 2000 times of consecutive fails to replace one of the parents.

5 Experimental Results

The melodic contours extracted by GAPTS are evaluated by SED. SED is the error in the string edit distance as mentioned in Section 4.2. Ten people hummed 95 songs for test. Following the convention for clear segmentation, they sang by "*ta*" or "*da*" into microphone under a usual condition with noise. All programs were written in *C++* language and run on PentiumIII 866MHz with the Linux operating system. We did not use any hardware device for acoustic signal processing except the microphone.

For robust comparison between Optus PTS and GAPTS (combined with GA), we followed the 5-fold cross-validation approach [17] [18]. We randomly divided the entire hummed query set D into 5 mutually exclusive subsets of approximately equal size. The GAPTS was trained and tested 5 times. Table 2 shows the sets of parameters; the first is the set of parameters that has been used in *HumSearch*, a commercial query-by-humming product of Optus Inc., and the second is the set we have found by the GA.

The original SED of *HumSearch* was 443. The GAPTS improved it to 424, which is a notable improvement. When we replaced the existing set of parameters in *HumSearch* by the new set, *HumSearch* also showed improvement. Out of the

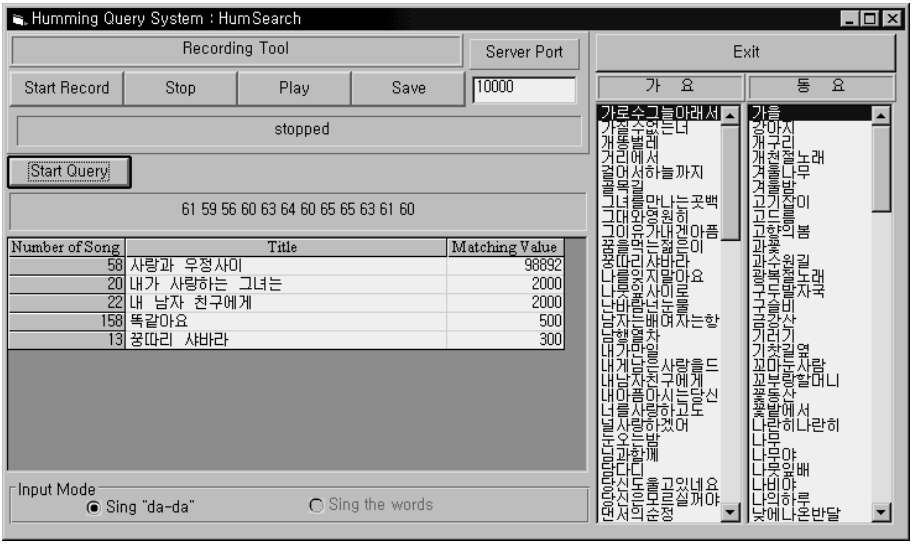

Fig. 12. A snapshot of our hummed query system

95 songs, we found 10 songs ranked up and 8 songs ranked down. We should note that the parameter set for the *HumSearch* has been tuned for a long time to be a commercial product. Considering this, the improvement in SED and the change of ranks is notable. Figure 12 shows a snapshot of our interactive hummed query system.

6 Conclusions

We extracted the parameters that control the note segmentation and pitch determination. We optimized them by combining the classical methods such as short-time energy diagram, zero-crossing rate, contour analysis of short-time energy diagram for note segmentation and cepstrum analysis for pitch determination. Our pitch tracking system worked well regardless of testing environments under various conditions, e.g., sexuality, noise, and input devices (mobile phone or microphone).

Acknowledgments. This work was partly supported by Optus Inc. and Brain Korea 21 Project. The RIACT at Seoul National University provided research facilities for this study.

References

1. Stephen Handel. Listening : An introduction to the perception of auditory events. In *The MIT Press*, 1989.

2. Ta Chou, Arbee L.P., and C.C. Liu Chen. Music database : indexing technique and implementation. In *Proc. of IEEE, International Workshop on Multimedia data base Menagement System*, 1996.
3. A. Ghais, J. Logan, D. Chamberlin, and B.C. Smith. Query by humming. In *Proceeding ACM Multimedia 95*, 1995.
4. Rodger J. McNab, Lloyd A. Smith, Ian H. Witten, Cclar L. Henderson, and Sally J. Cunningham. Towards the digital music library: Tune retrieval from acoustic input. In *Proc. ACM Digital Libraries*, 1996.
5. Sassan Ahmadi and Andreas S. Spanias. Cepstrum-based pitch detection using a new statistical v/uv classification algorithm. *IEEE Trans. Speech and Audio Processing*, 7(3):333–338, 1999.
6. Mingyang Wu, DeLiang Wang, and G.J. Brown. Pitch tracking based on statistical anticipation. In *Proceedings. IJCNN '01. International Joint Conference*, volume 2, pages 866–871, 2001.
7. L.R. Rabiner. On the use of autocorrelation analysis for pitch detection. *IEEE Trans. Acoust,, Speech, Signal Processing*, ASSP-25(1):24–33, 1977.
8. Ross. M, J, H.L. Shaffer, A. Cohen, R. Freudberg, and H. J. Manley. Average magnitude difference function pitch extractor. *IEEE Trans. Acoust,, Speech, Signal Processing*, ASSP-22(1):353–362, 1977.
9. Noll A.M. Cepstrum pitch determination. *Journal of Acoust, Soc, Amer.*, 41:293–309, 1967.
10. T. Shimomuro and H. Kobayashi. Weighted autocorrelation for pitch extraction of noisy speech. *IEEE Trans. Speech and Audio Processing*, 9(7):727–730, 2001.
11. Shimamura T. Kunieda N. and Suzuki J. Robust method of measurement of fundamental frequency by aclos: autocorrelation of log spectrum. In *IEEE International Conference Acoustics, Speech, and Signal Processing*, pages 232–235, 1996.
12. J.K. Kaiser. On a simple algorithm to calculate the 'energy' of a signal. In *Proceeding IEEE ICASSP*, pages 381–384, 1990.
13. D. E. Goldberg, K. Deb, and B. Korb. Do not worry, be messy. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 24–30, 1991.
14. Syswerda. Uniform crossover in genetic algorithms. In *International Conference on Genetic Algorithms*, pages 2–9, 1989.
15. D. Cavicchio. *Adaptive Search Using Simulated Evolution.* PhD thesis, University of Michigan, Ann Arbor, MI, 1970. Unpublished.
16. D. Whitley and J. Kauth. GENITOR: A different genetic algorithm. In *Proceedings of Rocky Mountain Conference on Artificial Intelligence*, pages 118–130, 1988.
17. B. Efron an dR.Tibshirani. Cross-validation and the bootstrap: Estimating the error rate of a prediction rule. In *Technical Report(TR-477), Dept. of Statistics, Stanford University.*, 1995.
18. R. Kohavi. A study of cross-validation and bootstrap of accuracy estimation and model selection. In *Proceedings of hte Rourteenth International Joint Conference on Artificial Intelligence*, pages 1137–1143, 1995.