Guitar Sound Analysis and Pitch Detection

Larry Wu
Center for Computer Research in Music and Acoustics, Stanford University
Email: larrywu@ccrma.stanford.edu

ABSTRACT

The pitch detection of music sound is one of the important technologies in audio signal processing. It can be used for interactive computer-music performance. In this paper, we consider some characteristics of guitar and proposed a specific and accurate pitch recognition method. Musical notations (MIDI files) are produced from the acoustic wave files. At the same time, the MIDI data will be sent to sound card to generate music sound.

1. INTRODUCTION

The process of transforming melody into a musical score is difficult to the beginners, although it is fundamental for the musical activities. So the development of a system that can take musical note automatically is practically very important and desired. Such a system is composed of various fundamental techniques, such as pitch recognition of musical sound, rhythm recognition, tempo recognition and musical instrument recognition [1]–[7]. After getting this information, we can develop many applications, such as auto-accompaniment system based on it. In our experiments, the instrumental recording wave file was carefully analyzed and extracted several appropriate parameters. Among those parameters, detecting and tracing the fundamental frequency (pitch) is one of the most difficult problems. This paper focuses on pitch recognition of the guitar sounds, and automatic notation transformation. The rest of the paper is organized as follows. In section 2, we describe how to determine the fundamental frequency in the frequency domain. In section 3, we discuss how to use the envelope in the time domain to find out one note's length. In section 4, we talk about the proposed system. Future works are discussed in section 5.

2. PITCH DETECTION ALGORITHM

Because we use the frequency domain information to estimate the fundamental frequency, the sample data in the time domain should be cut into many frames. The way to cut a frame is to shift *M* samples and select *N* samples data. *M* is the number of overlap of two frames. *N* is the frame size which is related to the precision in the frequency domain. As we know, the frequency interval between lower pitches is shorter than that between higher pitches. The pitches range of a guitar is around 82Hz (E2) ~660Hz (E5), so we need higher precision in the frequency domain to correctly determine the fundamental frequency. We set *N=4096* and *M=2048*. Then, we use following methods to find out the fundamental frequency in every frame.

2.1. Fast Fourier Transform (FFT)

The conventional Fast Fourier Transform (FFT) calculates the spectrum of the musical sound using the Hamming window given by

$$w(n) = 0.54 - 0.46\cos(2\pi n / N) \qquad n = 0,1\ldots N-1$$

where *N* is the Frame size.

2.2. Calculating Protrusion Value

After getting the frequency domain data, we should find out the magnitude of local peak. However, not every local peak is important. The more protrusive the local peak is, the more important it is. Usually, the small protrusion is caused by the noise signal. Therefore, we calculate the protrusion for every local peak and set a threshold to determine which local peak should be ignored.

$$protrusion = \frac{V_A}{Max(V_B, V_C)}$$

where V_A is the peak magnitude, V_B, V_C are the troughs magnitudes on both sides of the peak. If we directly calculate the peak's protrusion value, we will get some inaccurate results like Fig. 1.

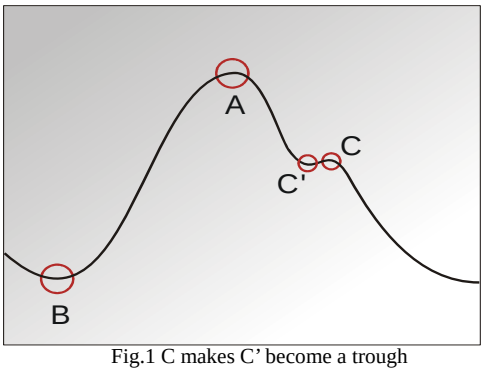

Fig.1 C makes C' become a trough

The little protuberance *C*, makes *C'* become a trough, and the *C'* may cause the protrusion of A not high enough to pass the threshold. For that reason, before calculating the protrusion, we apply a mean filter to all frequency samples to smooth this kind of protuberance. For guitar, 1.25 is a suitable value for the threshold.

2.3. Sorting Local Peaks by Protrusion Value

In general, the fundamental frequency and its harmonic frequencies have the strongest protrusion values. Therefore, we sort all local peaks by protrusion values and select the first *N* strongest peaks as the candidates of fundamental frequency. If *N* is too small, we may miss the fundamental frequency like Fig. 2.

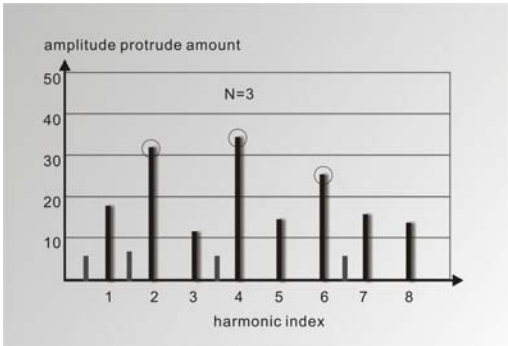

Fig.2 N=3, harmonic index 1 means fundamental frequency

If *N* is too large, we may get a non-harmonic frequency and it will cause a miscalculation of fundamental frequency like Fig. 3. Apparently, the *N* is related to the accuracy of pitch recognition result. For guitar, *N=4* is an appropriate value.

2.4. Quadratic Interpolation

The DFT is defined only for frequencies:

$$w_k = 2\pi k f_s / N, \qquad k = 0,1,\ldots N-1$$

where f_s is sample rate, and N is frame size.
When we use the DFT to analyze arbitrary signals from nature, it will cause *spectral leakage*. For that reason, we apply quadratic interpolation method to find out the more precise

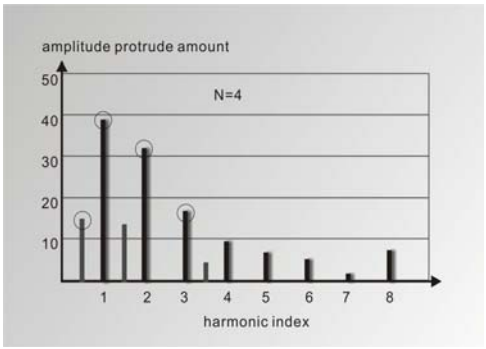

Fig.3 N=4, non-harmonic index is selected

place of the first *N* strongest frequencies. This method could be illustrated with the Fig. 4.

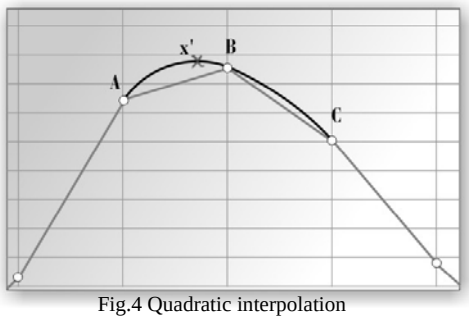

Fig.4 Quadratic interpolation

We desire a general parabola of the form:

$$y(x) = a(x - x')^2 + b$$

In Fig. 4, $A = y(-1), B = y(0), C = y(1)$
Solving for the parabola peak location, we get

$$x' = \frac{1}{2}\frac{A - C}{A - 2B + C}$$

The estimation of the true peak location will be

$$k^* = k_B + x'$$

and the frequency in Hz is $k^* f_s / N$

2.5. Greatest Common Divisor

Sometimes, the magnitude of guitar's fundamental frequency is lower than the magnitude of its harmonic frequencies. As a result, we only get *N* harmonic frequencies without fundamental frequency (Fig. 2). To solve this problem, we could calculate the Greatest Common Divisor among them. However, the frequencies' values are floating points, so we cannot calculate *GCD* by Method of Successive Division. So, we build a table (Fig. 4) which contains the basic relationship among numerator, denominator, and quotient (Fig. 5). From this table, we could know the most possible ratio of any two frequencies. The algorithm is (1) selecting any two frequencies, *A* and *B*, from the first *N* strongest frequencies. (2) if *A>B*, then *swap(A,B)*. (3) if *2*A>B*, then replace *A* with *B-A* (∵*GCD (A, B) = GCD (B-A, B)*) (4) set

$C=A/B$, and find out the i such that $(C- q_i) / q_i$ is the smallest value. q_i is the quotient value in the row i. (5) the *GCD (A, B)* will be B / d_i The d_i is the denominator value in the row i. (6) after calculating every *GCD* of any two frequencies from the first N strongest frequencies, we use a voting system to find out the real *GCD* among all first N strongest frequencies. (Voting system will be explained on 2.7)

Numerator	Denominator	Quotient
1	2	0.5000
1	3	0.3333
1	4	0.2500
1	5	0.2000
2	5	0.4000
1	6	0.1667
1	7	0.1429
2	7	0.2857
3	7	0.4286
1	8	0.1250
3	8	0.3750
1	9	0.1111
2	9	0.2222
4	9	0.4444
1	10	0.1000
3	10	0.3000
1	11	0.0909
2	11	0.1818
3	11	0.2727
4	11	0.3636
5	11	0.4545
1	12	0.0833
5	12	0.4167

Fig.5 Ratio table

2.6. Weight

Besides the *GCD*, we also calculate the *weight* which stands for the importance of this *GCD*. In Fig. 6, the magnitude of fundamental frequency (point A) is strong, but the magnitude of its harmonic frequency (point C) is small. On the other hand, there is a local peak in half fundamental frequency (point B) whose magnitude is higher than the magnitude of point C. In this situation, the *GCD* will be point B instead of point A. To avoid this problem, for every *GCD*, we give it a weight.

$$weight = \sqrt{MIN(V_1, V_2)}$$

where V_1 , V_2 are the magnitudes of two frequencies. If the magnitudes of both frequencies are strong, the weight will be strong too. It means this *GCD* is very important. If one magnitude is strong but the other is small, we choose the small one as the weight. It means this *GCD* is not very important. If the magnitudes of both frequencies are small, the weight will be small too. It means this *GCD* is least important.

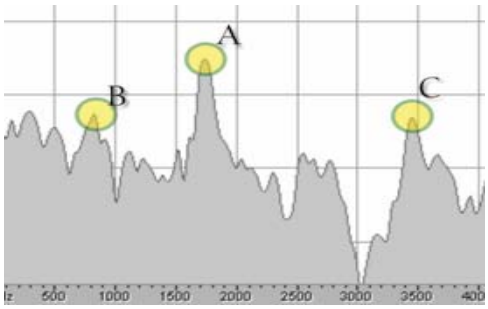

Fig.6 Point B's magnitude is bigger than C's magnitude

2.7. Voting System

The *GCD* is also a floating point value. Therefore, we quantize it by dividing a real number and round it off. This step groups two *GCD* values which are close to each other into one candidate. The voting system could be illustrated with Fig. 7, Fig. 8, and Fig. 9. In Fig. 7, we can see the Round Off values which are the candidates for *GCD*. In Fig. 8, *C[i]* is the candidate value. *V[i]* is the accumulated vote. *δ[i]* is the accumulated weight. Actually, the GCD value 83.4 should be in the same group of 82.2 and 81.6. However, "round off" separates it into another group. This is not what we want. So, if $|C[i]-C[j]|=1$, we add *C[j]*'s accumulated vote and accumulated weight to *C[i]*'s accumulated vote and accumulated weight (Fig. 9).

GCD (G)	Quantize Factor (Q)	G/Q	Round Off
82.2	2	41.1	41
83.4	2	41.7	42
164.8	2	82.4	82
81.6	2	40.8	41

Fig.7 Quantization

i	*C[i]*	*V[i]*	*δ[i]*	(GCD value, weight)
1	41	2	7	(82.2, 4) (81.6, 3)
2	42	1	4	(83.4, 4)
3	82	1	5	(164.8, 5)

Fig.8 Voting System

i	*C[i]*	*V[i]*	*δ[i]*	(GCD value, weight)
1	41	3	11	(82.2, 4) (81.6, 3) (83.4, 4)
2	42	1	4	(83.4, 4)
3	82	1	5	(164.8, 5)

Fig.9 Revision

The bigger the accumulated weight *δ[i]* is, the more possible the fundamental frequency is among the *C[i]*. After sorting all candidates by accumulated weight, we find out the *i* for which *δ[i]* is the largest value. Then, we calculate the median point of all *GCD* value in *C[i]* by

$$r = \frac{\sum_{k=1}^{N} \delta_k GCD_k}{\sum_{k=1}^{N} \delta_k}$$

where r is the median point, N is the number of votes, and δ_k is the accumulated weight. The r could stand for the fundamental frequency of this frame. After getting the fundamental frequency, we convert it from Hz to Midi number using following formula

$$m = \log_2(\frac{f}{261.62}) \times 12 + 60$$

where m is Midi number and f is frequency.

3. NOTE LENGTH DETECTION ALGORITHM

The envelope of the guitar sound is a standard ADSR envelope (Fig. 10). With this character, we can easily cut a section for every note. These methods are discussed in the following steps.

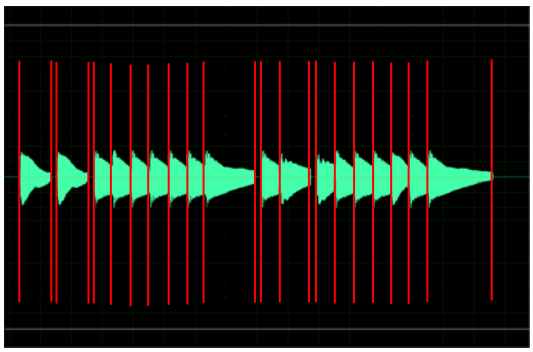

Fig.10 Envelope in time domain

3.1. Cut Section

In section 2, the time domain data has been cut into many frames. Every frame has its representative frequency. Besides, we calculate the average of all samples' amplitudes in this frame. This average amplitude could stand for the volume of this frame. We use the same way discussed in section 2.2 to find out the local volume peaks and troughs among all frames. First, a mean filter is applied to the envelope to smooth its shape. Second, we calculate the protrusion for every local volume peak and set a threshold to determine which pick should be ignored. Moreover, we set another threshold to neglect the frame whose volume is too small. That is because the signal with too small amplitude cannot be heard, and it is like a noise signal. One local volume peak and two troughs could define a section of a note. In Fig. 10, every red line is the place of a trough. Every two redlines with a local volume peak between them is a section of a note.

3.2. Error Correction

In previous step, if the local volume peak of an existed note is not greater than the threshold, it will be neglected and be classified to other note's section. To solve this problem, we use the voting system again. This voting system could be illustrated with Fig. 11, Fig. 12, and Fig. 13. As we know, there are many frames in one section, and every frame has a representative frequency. We define that a single frequency's weight is 1. The successive frequency's weight is 3. The third successive frequency's weight is 6, and so on. There are two thresholds value to determine if there is another note in this section. One is for the note's weight, and the other is for the weight ratio between two notes. For example, the largest weight in Fig. 13 is 19, and the second one is 8. Suppose the weight threshold is 7 and the weight ratio threshold is 0.3. Weight 8 is greater than the weight threshold 7, and the weight ratio between weight 8 and weight 19 is 0.42 which is greater than 0.3. So, the second frequency 44 should be separated from this section and become another section.

Every frames' frequency in one section									
44	44	22	44	44	88	88	88	88	22

Fig.11 The frequency is Midi Number

Every frequency's weight									
1	3	1	1	3	1	3	6	9	1

Fig.12 Weight

Freq. candidate	Total weight
44	1+3+1+3=8
22	1+1=2
88	1+3+6+9=19

Fig.13 Voting System

3.3. Get Pitch and Volume

In every section, we choose the frequency candidate with largest weight as this section's frequency, and we choose the largest volume among all frames as this section's volume.

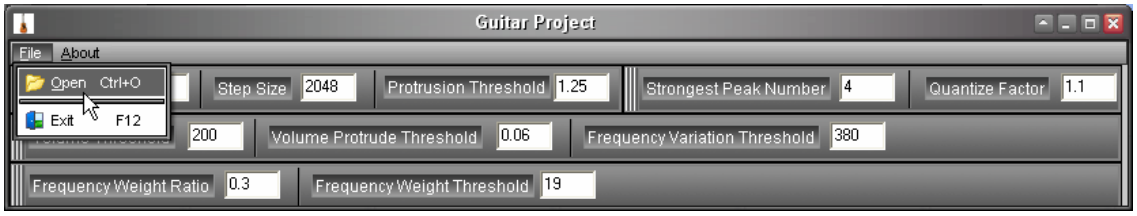

Fig.14 Main program

4. IMPLEMENTATION

We use C++ to implement the algorithm discussed above. The parameters mentioned in the algorithm could be directly changed by users. The default values are suitable for the guitar sound. For other instruments, we need different sets of parameters. This program and the five test wave files are in the same directory. You could listen to the test wave file first. Then, you use this program to open the same test file. This program will analyze the file and recognize the pitch and volume. After getting the pitch and volume information, this program converts them to Midi message and sends them to sound card to generate music sound. At the same time, it saves a Midi file in the same directory. You also could listen to the Midi file to compare the analysis result to the original file.

5. FUTURE WORK

What we want to do next is convert the recording wave files or real-time performance to the SASL files. The analyzer will be suitable not only for the guitar, piano, flute, but also for other Chinese musical instruments, like the erhu, pipa, and dulcimer. It will be an instrument independent algorithm. Moreover, the analyzer can extract the expression information which makes the music alive and have personality. Expression is related to the player and we want the synthesizer to be a player not a machine. It is an add-on to the existing synthesis method. For example, guitar has the expression like staccato, portamento, and vibrato. This synthesizer can have its own style and generate style based on the song and imitate styles from for example, Eric Clapton, or Miles Davis. This is why MIDI is not enough and we need JavaOL [8][9]. Besides, when the synthesis algorithm in the JavaOL files generates the sound, the computer will automatically accompany it on the other musical instruments in real time. After we have done that, we want to apply the surrounding effect, which will first be realized on headphone, to the final sound. Then, I will simulate this effect on two speakers. Ultimately, I will implement this effect on the up-to-date technology, audio spotlight, by which the highly directional wave could be produced. That way, the surrounding effect could be made precisely by one audio spotlight speaker.

6. REFERENCE

[1] Sterian, A., Simoni, M. H. and Wakefield, G. H., “Model-Based Musical Transcription,” In Proceedings of the 1999 international computer music conference, 1999.

[2] Godsill, S. and Davy M., “Bayesian Harmonic Models for Musical Pitch Estimation and Analysis,” IEEE International Conference on Acoustics Speech and Signal Processing, 2002.

[3] Maher, R., “Fundamental frequency estimation of musical signals using a two-way mismatch procedure,” J. Acoust. Soc. Of Am., Vol. 95, no. 4, pp. 2254–2263, April 1994.

[4] Selfridge-Field, E., “Optical recognition of music notation,” A survey of current work, Computing in Musicology: An International Directory of Applications, Hewlett, W. B. and E. Selfridge-Field (eds), Thomas-Shore, Michigan, 1994, vol. 9, pp. 109–145.

[5] Bainbridge, D. and N. Carter, “Automatic Recognition of Music Notation,” Handbook of Optical Character Recognition and Document Image Analysis, H. Bunke and P. Wang (eds), World Scientific, Singapore, 1997, pp. 557–603.

[6] Tadokoro, Y and Yamaguchi, M., “Pitch Detection of Duet Song Using Double Comb Filters,” Proc. of ECTD’01, Vol.I, pp. V- 57–60, 2001.

[7] Kajiyama, K and Yamashita, Y., “Pitch Recognition of Musical Sounds Based on Comb Filters and a Stochastic Model,” IPSJ SIGNotes MUSic and computer Abstract, No. 044-009, pp. 53–58, 2001.

[8] Alvin W.Y. Su, Yi-Song Xiao, Jia-Lin Yeh and Jian-Lung Wu, “Real-Time Internet MPEG-4 SA Player and the Streaming Engine” accepted and published in Audio Engineering Society the 116th Convention, 2004

[9] Yi-Song Xiao, Alvin W.Y. Su, Jia-Lin Yeh and Jian-Lung Wu, “A Structured Audio System Based on JAVAOL” accepted and published in Workshop on Computer Music and Audio Technology, Taiwan, 2005