Appear in Proceedings of 1997 International Symposium on
Multimedia Information Processing, 1997

Generating Guitar Scores from a MIDI Source

Jeng-Feng Wang and Tsai-Yen Li[*]

Computer Science Department
National Chengchi University, Taiwan, R.O.C.

Abstract

Most music-related software packages were designed for students to learn music theories or for the music professionals to compose music. However, amateur music instrument players often need another form - acquiring an instrument-specific score from a music source. In this paper, we describe a software package capable of automatically generating guitar chords and fingering styles from music sources in MIDI format. The software program extracts melody from a MIDI file, uses weighted rules based on harmonics to select appropriate chords, and then uses principles of guitar composition to generate a six-line score. This program, developed under MS Windows environments, also allow users to interactively edit chords and fingering styles in order to create a customized guitar score.

1. Introduction

Computer music has been an active field for several decades. Most software applications developed for computer music have been to assist music composers to create synthetical music. Creating a good piece of music not only requires creativity but also a solid background on music theories. However, for most novice players of musical instruments, who may not have rigorous music training, this kind of software may not be very useful. On the reversed, what amateur players need often is a service that converts a song or a piece of music into an instrument-specific score that they are familiar with. A typical example would be to convert a song into a six-line score with appropriate fingering styles for guitar players.

The main problem in formalizing such a process is on mapping notes into a chord, which is not a one-to-one mapping for a measure. Good choices of chords often require context information like in natural language processing. Although there

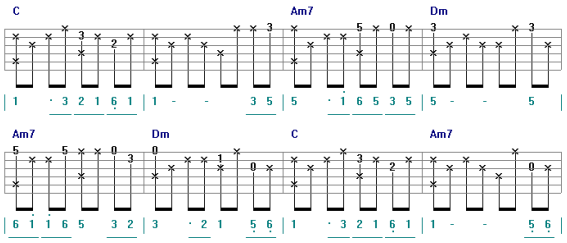

Figure 1: A typical guitar score

are rules to follow in harmonics, there is no formal definition for how good a set of chords is. It is more likely that the criteria for such a judgement will need to refer to personal perception eventually. Only experienced musicians can tell good matches from bad ones, and the experience is a black art that is difficult to quantify.

In this paper, we describe a software program aiming at automating this score creation process from a raw music source. Specifically, the software takes a MIDI (Musical Instrument Digital Interface) file [1], converts it into a simple score, matches the score with the "best" chord set, and then produce a six-line score for guitars as shown in Figure 1. Users are allowed to intervene the process by modifying immediate or final outputs. In particular, a user can interactively modify the chord and fingering style for any measure to create a more harmonic match in his/her taste or simply to add variation in chord progression.

The rest of this paper is organized as follows. In Section 2, we describe previous researches in computer music pertaining to our work. In Section 3, we describe requirements for such a software application and the overall structure of our system. In Section 4, we give a more detail account on how each conversion step is accomplished. In Section 5, we present the results produced by our software package and report its performance. Section 6 concludes the paper with some discussions on possible future extensions.

[*] {s8239,li}@cs.nccu.edu.tw

2. Related Work

Most researches in integrating computer technologies and music theories fall into two categories: *computer-assisted music composition* and *computer-assisted music analysis*. The results obtained in these two domains are mainly used in music production and education.

The development of computer-assisted composition started as early as 1955 when researches on computer synthetic voice was initiated by Hiller and Isaacson of Illinois University and other researchers in Paris and Holland. Since then, computer-assisted music composition has played an important role in music creation. It also inspired the development of related software for theory learning in music education [10]. Typical contents of a computer-assisted course on theory learning include large staffs, notes and rests, intervals, time signatures, scales, key signatures, modulation, chords, measures, staff writing practices, music terminology, and so forth [2][4][8][9]. As computer hardware becomes more accessible, commercial software packages for music composition and learning also become more popular. For example, the Apple Computer Inc. developed a software package for children to learn music composition and theories. Features of this package include chord matching, Rondo and Canon writing, etc. [7].

Computer-assisted music analysis is another major field in music education. There has been many courses developed using this approach. These courses usually cover voice crossing, harmonics intervals, traid, Youngblood' s Hindeminthian Analysis of Root Progressions, Bartok's Method of Folk-Song Analysis [5], and Hormony Analysis. Blombach and Poland of Ohio State University also developed a course on the analysis of Bach Choral.

Despite the extensive use of computers in music education and digital music production, not much work has been done (at least not on every aspect) for amateur music players. The amateurs may not have rigorous training on music theory but still want to have access to playing instructions for recently published songs or music. For example, an amateur guitar player would greatly appreciate that computers can automatically translate his/her favorite songs into a guitar score that they can play on immediately. In the following section, we will point out the necessary components for such an application and describe how we design such a system.

3. The System Description

The software system that we have developed takes a MIDI file as input and generates a guitar score for it as output. The conversion process consists of the following three steps:

3.1. Melody extraction: In this step, we extract the music melody from a MIDI source and output it in a simple score format with appropriate bar lines attached. It is straightforward to extract such information from a MIDI file. In fact, several commercial staff-drawing software packages have made this capability as one of their features. Therefore, we do not describe this process in details. For more information on this topic, see [1]0.

3.2. Chord matching: The goal in this step is to select appropriate chords for the simple score using an appropriate sampling frequency. The main principle for the selection is to match the melody of the music with a "best-fit" chord. However, matching is not the only criteria for chord selection since it also depends on the context of adjacent chords. There are guidelines in music theories that one can follow to resolve dissonance in chord progression. Nevertheless, it is still difficult to have a systematic way to compute them since there are no dictatorial rules to match chords with scores.

In this paper, we propose a novel approach to find "best-fit" chords. In this approach, we evaluate each possible chord with weighted guidelines extracted from chord progression theories. All reasonable matches are selected initially as candidates, and each guideline is given a weight according to its importance in practice. Then the goal of the system is to narrow down the candidate set by applying these weighted guidelines in several consecutive steps. Only when a step results in multiple chords with equal weights, the next step is taken to resolve the ambiguity. The design principles used in this process are described as follows. More details will be presented in the next section.

a. Sampling frequency: Since the number of chords that can be put in a measure is not fixed, we first need to decide on how long we should sample the score in order to fill in appropriate chords. In our system, this decision is mainly based on the occurrences of different note periods in each measure. For example, the sampling accuracy is up to a eighth note (e.g.,

T1213121, etc., for guitar fingering styles) for a song in four-four time (4/4). If there are too many sixteenth notes in a measure, then half of a measure may be a more adequate sampling period.

b. Melody matching: Chords result from the resonance of melody notes. Thus, matching melody is the main principle in chord selection. Every note needs to be considered but dominant notes should take precedence. For instance, if the duration of a note is longer than half of the sampling period, the chords matching this dominant note should be given more weights. In addition, simplicity is also another practical consideration for chord selection. Therefore, if the melody does not contain sharp or flat notes, the chords without sharps and flats are preferred.

c. Chord progression: If multiple candidate chords result from applying the previous principle, the context of adjacent chords is the next to consider. In particular, we consider the strong connection principles in chord progression such as a root note fifth downward, a third downward, a second upward, and the golden chord principles. In addition, resolution of dissonance intervals also helps in making the decision.

d. Lowest notes: If none of the principles above help in choosing a unique "best-fit" chord, matching the lowest note of all voices in a measure is another good principle to follow. This is because the lowest note is usually the most obvious sound in a measure [3].

3.3. Guitar score generation: In this step, we use the melody and chords found in the previous steps to draw a six-line guitar score. Since the score is specific to guitars, common playing practices for guitars need to be accounted for. For example, melody notes are preferably raised by an octave when incorporated into fingering styles, and most finger positions are below the fifth cell simply for playing convenience.

In addition to the above design principles, the following functions on user interface are necessary for a complete application.

- Since the chord selection is not unique, the system needs to provide an interface for a user to change chords, as they desire.
- The system also needs to provide an interface for a user to change fingering styles for each measure to avoid monotonousness.
- In order to instruct the players how to follow the score, the system needs to provide a function that can play the music and highlight the note being played in real time according to the tempo of the music.
- The system should also provide nice printing capability so that the user can get a hardcopy of the final score.

All of these functions are part of our implemented system. In the next section, we will give a more detail account on how we implemented such a system.

4. Implementation

We will present our implementation in three steps as described above.

4.1. Melody extraction: By analyzing the channel message in a MIDI file, we only choose the main channel for processing. In the main channel, we use the "Note Off" and "Note On" of the 8nh and 9nh fields and their corresponding running statuses to decide the starting and ending times of each note. Then we use these two time stamps to calculate the length of each note. In addition, we use the 51h, 58h, and 59h fields of the FFh commands in channel messages to set up bar lines, tempo, and major key, respectively, in the simple score.

4.2. Chord matching: The rules that we follow to select an appropriate chord are depicted in the flow chart shown in Figure 2 (on the next page). Weights in units of points are used to evaluate the fitness of a chord. Chord selection is done in three substeps. If any substep results in a unique chord of the heaviest weight, no further criteria need to be applied, and the subsequent substeps are ignored.

In order to decide on a sampling frequency, we sample the whole music and count the number of occurrences of the eighth notes and the sixteenth notes. If the number of sixteenth notes is two times more than the number of eighth notes, half of a measure is used as the sampling period; otherwise, a measure is used as the default.

4.2.1. Basic screening: Three criteria based on melody are used to screen out obviously inappropriate chords.

a. Candidate chords: The initial weight for a chord is calculated based on the number of notes that match any of the constituent notes in

the chord. The more matches to a chord, the more points are given to it.

b. Dominant note: Based on the initial weight mentioned above, we add one point to each chord that contains the dominant note in the sampling period. If the length of the dominant note is longer than half of the sampling period, we add an extra point to its weight to emphasize its dominance.

c. Sharp and flat notes: If there are no sharp and flat notes in a sampling period, sharp or flat chords in the candidate set (if any) are discarded. On the other hand, chords without sharp and flat signs are discarded if the melody notes do have any of them.

If the above yields a chord that is uniquely the heaviest in the candidate set, then this chord is chosen for the final guitar score; otherwise, the following supplemental substeps are taken to resolve the ambiguity.

4.2.2. Second screening: We use principles in chord progression to further differentiate the best-fit chords that are found so far.

a. We give extra two points to the chords that are a fifth downward, a third downward, or a second upward with respect to the root note of each chord chosen in the preceding sampling period.

b. Two extra points are granted to those chords that do not fall into the above category but do follow common rules in chord progression (such as I→III or VI→IV).

c. If the previous chord is a dominant7 chord, seven special rules in resolving dissonance are applied to the candidate chords. For example, two extra points are given to those chords that make smooth transitions from the preceding chords such as I7→IV, II7→V, VI7→II, and so on. There are cases where a chord plays a transitional role in chord progression. In order not to exclude this possibility, some credits are given to these transitional chords. For example, V7 can transit to V via II7. In this case, one extra point is given to the transitional chord.

If applying the above criteria to the candidate chords yields a unique chord of the heaviest weight, the chord is chosen for the final score; otherwise, we proceed to the next substep to resolve the ambiguity.

4.2.3. Final screening: One final auxiliary criterion is on the lowest note. If the lowest note of a sampling period is the root note of a candidate chord, then two extra points are given to the chord. If this final weighting can not differentiate two equally best-fit chords, then the choice is arbitrary.

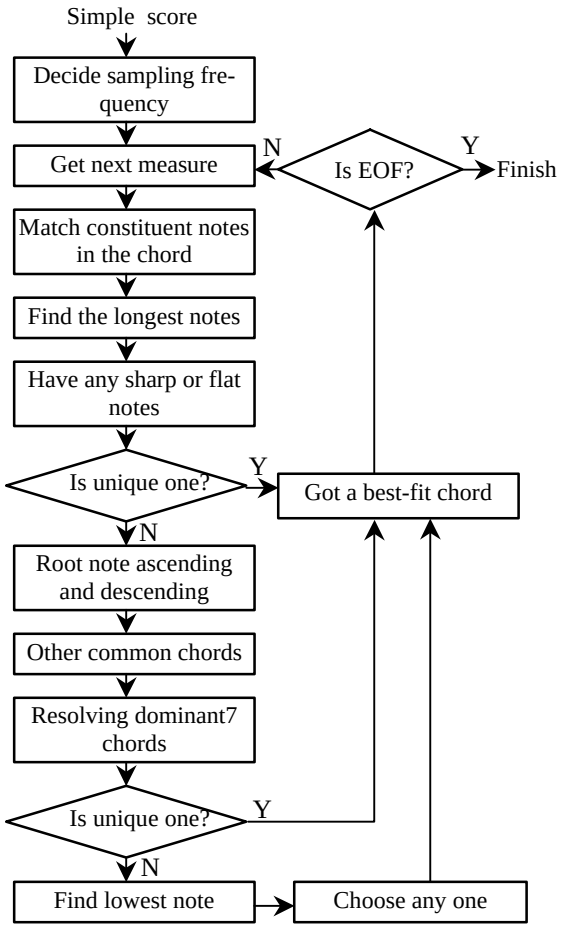

Figure 2: Flow chart for selecting chords

4.3. Guitar Score Generation: The following rules are used to generate a guitar score that contains adequate fingering styles.

- Choose a primary fingering style according to the rhythm and raise melody notes by an octave. Whenever possible, the raised melody notes are incorporated into the primary fingering style.
- If the primary fingering style conflicts with the melody notes, melody notes take precedence.
- If the finger position for a raised melody note is higher than the fifth cell and the current string is not the highest string, wrap it to the next higher string.

Figure 3: Snapshot of the user interface

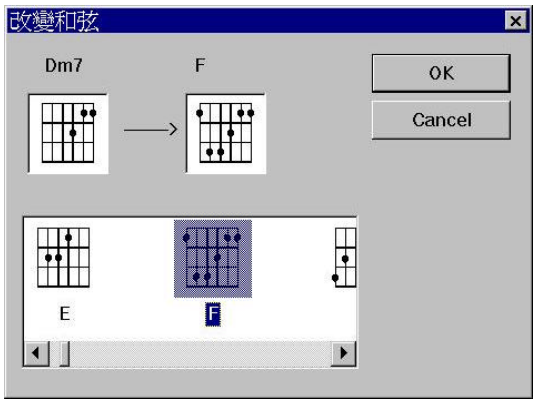

Figure 4: Interface for changing a chord

- If the raised note is over the twelfth cell on the highest string, lower the note to its original position.

4.4. User intervention: A reasonably good guitar score can be generated automatically using the above process. However, some user intervention may further improve the resulting score. For example, although chords are generated based on rules in music theories, there could be other chord sets that may sound better than the one selected by the system. In addition, it may be more desirable to increase the variation of fingering styles or variation on local sampling frequency. Therefore, we design an interface that allows a user to modify intermediate or final results interactively in each of the aforementioned steps to correct or improve the overall score.

5. Experimental Results

5.1 User interface

The software system[1] was implemented in Visual C++ for 32-bit Windows operating systems. A snapshot of the user interface for a typical song is shown in Figure 3. Figure 4 shows a dialog box that allows a user to change a chord interactively. According to the tempo of a song, the system can also animate fingering styles in real time by changing the colors of corresponding picking symbols. In addition, the program can produce a nice hardcopy of the guitar score for offline uses.

5.2 System performance

Some experiments have been conducted to observe the effects of different chord selection rules and to measure the overall system performance. The experimental results are summarized in Table 1 (on the next page). The data are taken on a 486-DX66 personal computer for three typical MIDI files. The number of measures in these MIDI files ranges from 42 to 78. More than half of the chords were uniquely decided after the first screening while the others were determined in the second and the final screening. The average number of chords left undistinguishable (after the final screening) ranges from 2.04 to 3.5. The choice for these chords is arbitrary. These results show that the selection rules are adequately weighted. The multi-step screening strategy does save processing time by letting a good matching chord stand out in early stages. The running time for the overall process is only fractions of a second, which is good enough for an interactive application.

6. Conclusion

For all music instrument players, it is highly desirable to have a computer automatically translate the music that they have heard into scores that they can play with. Using music theories in harmonies and in guitar composition, we have implemented a software system that converts MIDI music into a six-line score for guitar playing instruction. The guitar scores generated by the program are all quite satisfactory in common practices. We hope that this software can be beneficial to all amateur music players as well as the professions.

In our program, we assume that the input file is a well-constructed MIDI file in order to simplify the process of melody extraction. In practice, most music sources are not in MIDI format yet. It

[1] The system is available for anonymous ftp at `ftp://www.cs.nccu.edu.tw/pub/li/packages/genchord.zip`.

Example #	1	2	3
File Size(Bytes)	22605	13800	19300
Num. of Measures	59	78	42
Sampling Freq. (chords/measure)	2	1	1
Basic Screening	63(54%)	41(53%)	22(52%)
Second Screening	13(11%)	13(17%)	6(14%)
Final Screening	42(35%)	24(30%)	14(33%)
Avg. Num. of Chords Left	2.85	3.50	2.04
Running Time(sec)	0.38	0.27	0.16

Table 1: Experimental data for three examples

would certainly be more desirable if the system can take its input directly from analog sources such as CD audio. However, extracting melody from natural sound is a challenging task and is beyond the scope of this paper. Nevertheless, except for the initial melody extraction, the same processing steps can be reused for applications with such extensions. In addition, the results would be more fruitful if this work is extended to produce scores in other special forms for various music instruments.

Acknowledgement

This work was partially supported by a grant from NSC under contract number NSC 86-2213-E-004-006. The authors also would like to thank Hui-Chi Wu for her careful review of this paper on music terminology.

Bibliography

[1] S. De Furia, and J. Scacciaferro, "*The MIDI Programmers's Handbook*," Redwood City, Calif, M&T, 1989.

[2] F. T. Hofstetter, "Foundation, Organization, and Purpose of the National Consortium for Computer-Based Musical Instruction," *Journal of Computer-Based Instruction*, 3, pp21-33, 1983.

[3] J.-Y. Huang and S.-W. Pan, "*Music Wizards - Breakthrough in Guitar Playing*," I-Ping publishing, Taiwan, pp82-83.

[4] Y.-M. Sheu, "*On Computer-Assisted Music Education - Analyzing the Effects of Recognition Learning on Notes and Rests*," Masters thesis, Music Department, National Normal University, Taiwan, June, 1993.

[5] B. Suchoff, "The Computer and Barok Research in America," *Journal of Research in Music Education*, 9(1), pp3-16, 1971.

[6] M.-H. Tsai, "*On Extracting Melody in Multi-Channel Music*," Master thesis, Department of Computer and Information Engineering, National Taiwan University, Taiwan, June, 1996.

[7] R. Upitis, "A Computer Assisted Instruction Approach to Music for Junior-Age Children: Using ALF for Teaching Music Composition," in *Proceedings of the International Computer Music Conferences*, April 1982.

[8] R. Upitis, "Milestones in Computer Music Instruction," *Music Educator's Journal*, 48(04A), pp865, 1983.

[9] Gerhard Widmer, "Qualitative Perception Modeling and Intelligent Musical Learning," Computer Music Journal, 16(2), pp51-68, 1992.

[10] Y.-H. Young, "*On Computer-Assisted Music Composition Teaching*," Master thesis, Music Department, National Normal University, Taiwan, June 1993.