Automatic Guitar Music Transcription

Lance Alcabasa
College of Computer Studies
De La Salle University
Manila, Philippines
lancealcabasa@yahoo.com.ph

Nelson Marcos, PhD
College of Computer Studies
De La Salle University
Manila, Philippines
nelsonmarcos2003@yahoo.com

Abstract—**This paper presents a system that helps in automatically generating guitar tablatures and musical scores based on musical audio data. Information gathered from the audio consists of pitch, onsets and durations, chords, and beat and tempo. Major issues that were encountered during the research were harmonics for pitch detection, thresholding for onset detection, chord distinction, similar chord structures for chord labeling, and the subjective quality of tempo. Results were generally acceptable, performed on a data set that contains 22 files with varying elements. 70% accuracy was gathered from pitch detection, 60% accuracy from onset detection, 86% accuracy for chord distinction, 85% accuracy for chord labeling, and 81% accuracy for beat and tempo.**

Keywords- music transcription, signal processing, frequency domain, tablature, score, chord, onset, pitch, beat, tempo

I. Introduction

Written music or music transcriptions are physical representations of music, similar to how words are translated into text. They exist various forms. Two of these are scores and tablatures. Scores or sheet music are generally considered as the standard way to transcribe or write music. A musical score can be applied to a wide variety of instruments. Tablatures on the other hand are written music mainly for guitars. Each form has their own advantages and disadvantages, but regardless, both are created to help musicians learn, play, and share a particular piece of music. There are many useful technologies and systems available that focus on this particular aspect of music, such as research done by [1], [2], [3], [4], [5], [6], and [7].

There are many commercial software which contain features that can create and play musical scores, and some of them even have MIDI to music score converter. Examples of this software include Guitar Pro and Teach Me Piano.

Creating a music transcription itself can be a tedious task for people. Although there exist systems for creation of transcribed music, almost all generate musical scores, lacking a focus for the guitarist community. As such, the aim of this paper is to develop a system that will focus on the automatic generation of guitar tablature.

II. Framework and Modules

The framework for this research is shown in Fig. 1. The input is in the format of a .WAV file, which either contains recorded guitar music from a microphone, or pre-recorded guitar music from an audio CD.

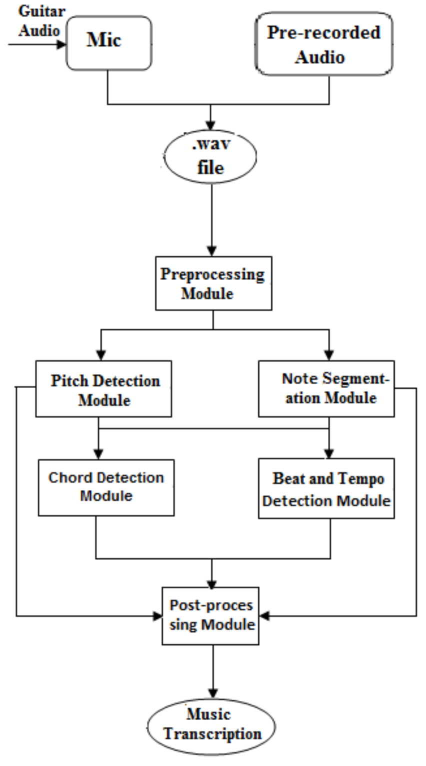

Figure 1. Architectural Framework

The audio file is then passed on to the succeeding modules, which are the pre-processing module, pitch detection module, note segmentation module, chord detection module, beat and tempo module, and post-processing module.

A. Pre-processing Module

In this module, the running-pass and band-pass filters were implemented to be able to gather better results.

The running-pass filter is a simple filter which works on the time-domain of the audio signal. The running-pass selects values of the signal by blocks, resulting in a cleaner signal, but also possible loss of data.

The band-pass filter is a simple filter which works on the frequency-domain of the audio signal. Using the band-pass filter allows selective use of frequency-range in the frequency domain (similar to performing both a low-pass and high-pass filter). By doing this, frequencies that are not relevant for use in the research are disregarded.

B. Pitch-Detection Module

Pitch detection deals with detecting various pitch at various points in time, mainly the pitch name and their corresponding octave. Two algorithms were implemented for this module, which includes using Fast Fourier Transform and Auto-Correlation. An overview of the algorithms can be found in [8].

The first algorithm makes use of the frequency-domain of the signal. The frequency-domain, shown in Fig. 2 [8], can be achieved by applying the Fast Fourier Transform to the time-domain of the signal.

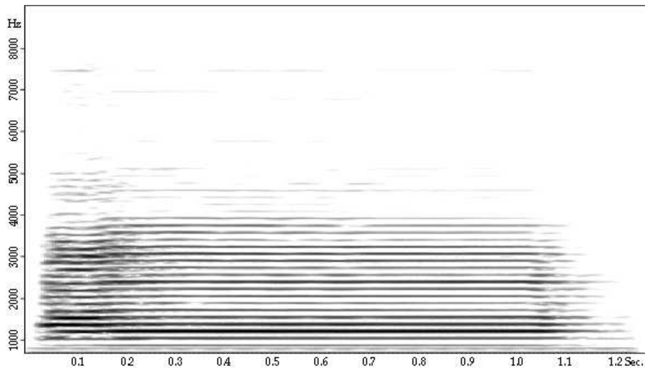

Figure 2. Frequency Domain Representation

The frequency-domain contains information on the different intensities of various frequencies at different points in time. In Fig. 2, the x-axis represents time, the y-axis represents frequency, and the z-axis (which corresponds to how dark a line is) represents the intensity.

To obtain the pitch from the frequency-domain, the fundamental frequency is calculated from the frequency-domain. There are algorithms to get this such as Harmonic Product Spectrum, due to problems regarding quality of sound. However, since this research focuses on guitar which is a percussive instrument, a simple approach was used to obtain the fundamental-frequency, which is to get the frequency bin with the highest intensity, as shown in Fig. 3.

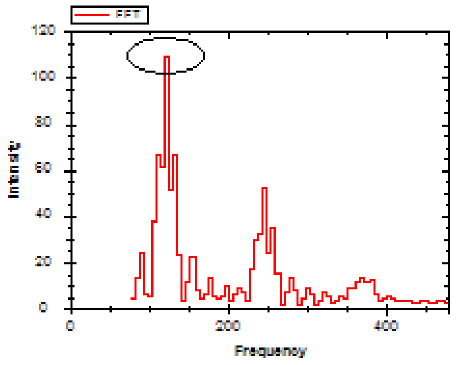

Figure 3. Fundamental Frequency

The second algorithm for pitch detection is auto-correlation. Since a periodic signal will eventually repeat itself, the concept of auto-correlation is to get the fundamental frequency based on cross-correlating a signal with itself. As the offset of the cross-correlated signal nears the fundamental period, the correlation of the two signals will become better, as shown in Fig. 4 (the blue signal being the original, and the green being the cross-correlated signal). The offset with the largest correlation value is considered the fundamental frequency.

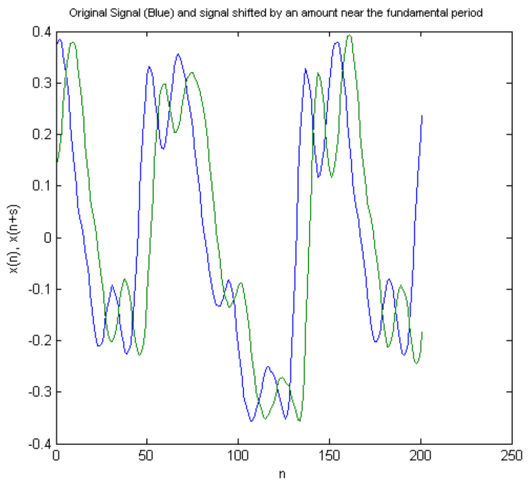

Figure 4. Auto-correlation

C. Note-Segmentation Module

The note segmentation or note onset detection deals with segmentation of note onsets or detecting when a new note is played in the audio. This module follows the general framework by [9] shown in Fig. 5 [9], which includes pre-processing, detection function, and peak-picking.

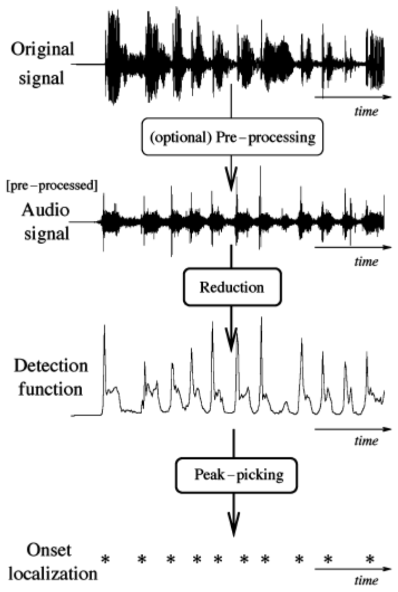

Figure 5. Onset Detection Framework

As discussed, filtering algorithms were done for the pre-processing of the signal. The detection function, or reduction aspect of note onset detection deals with enhancing the signal such that peaks representing new note onsets are emphasized. This is done by using either the time-domain or the frequency-domain (or possibly even both). The algorithm for detection-function is the spectral flux, which works on the concept of frequency-content difference. The algorithm for spectral flux is shown in equation (1), where k is the kth spectrum or the kth FFT block of the signal, s(k,i) is the value of the ith bin of the kth spectrum, and s(k-1,i) is the value of the ith bin of the previous spectrum.

$$SF(k) = \sum_{i=0}^{n-1} s(k,i) - s(k-1,i) \tag{1}$$

Lastly, peak-picking works with detecting which peaks are to be considered as note-onsets. This is mainly done through thresholding. However, since there are various problems in working with static thresholding, this research used a dynamic thresholding algorithm (running-mean), which is also from the paper of [9]. The idea is to set local-thresholds for each point, as shown in Fig. 6. The formula for running-mean used in this research is shown in equation (2), where n_i is the current value, w is a window size, and M is a constant multiplier.

$$locthresh(n) = \frac{1}{w} * \sum_{i=n-(w/2)}^{n+(w/2)} n_i * M \tag{2}$$

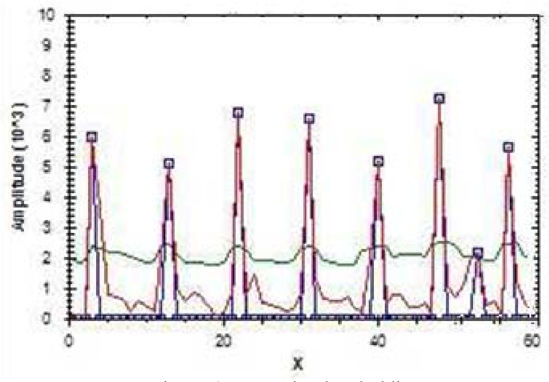

Figure 6. Dynamic Thresholding

D. Chord Detection Module

The chord detection module deals with detection and labeling of chords. This module is divided into two sub-categories which are the chord distinction and chord labeling. Chord distinction deals with distinguishing single-notes from chords while chord labeling deals with assigning chord labels to distinguished chords.

For chord distinction, spectral content are used as the basis for distinguishing single-notes and chords. This is based on the theory and observation that since chords are made up of multiple-notes, chords also have generally higher frequency content compared to single-notes.

For chord labeling, chroma vectors were used for determining the chord structure. Chroma vectors are essentially arrays that contain information regarding the intensity for different pitch [10]. An example of a chroma vector for the C Major chord, which contains the notes C, E, and G, is shown in Fig. 7.

1	0	0	0	1	0	0	1	0	0	0	0
C	C#	D	D#	E	F	F#	G	G#	A	A#	B

Figure 7. C Major Chroma Vector

E. Beat and Tempo Module

The beat and tempo module deals with detecting beats as well as estimating the tempo (which is measured in beats-per-minute). The beat is defined as a point-in-time that occurs at a regular interval in the music. Most of the musical events (such as the beginning of a new note) fall on the beat or subdivisions of the beat (half-beat, quarter-beat, etc.). The algorithms used for this research include sound-energy peaks, pulse-trains, and a library developed by Queen Mary, University of London, which uses a combination of the Two-State Beat Model [11] and [12].

Working on the concept of sound energy peaks is that beats are generally the strongest peaks in the music, especially for music including percussions where the bass drum, snare, or hi-hat which all produce strong percussive sounds signify the beat. [13] Basically, this is similar to detecting note onsets, in which peak-picking is generally done to determine which peaks to be considered as beats.

The next algorithm considers the use of pulse-trains, as illustrated in Fig. 8 [13]. The concept of pulse-trains is that one of the ways to extract when beats occur is to observe the note onsets using a predefined tempo. [14] Note that for each tempo, the beats are found on different points in time. Using the predefined tempo, compare the beat locations from the predefined tempo to the current data. This is done by starting from the first note onset and adding the sample beat interval (which comes from the predefined tempo). If a note onset is found after comparison, then the sample tempo maybe the correct tempo. It is also possible to explore other tempos based on the performance of the previous sampled tempo.

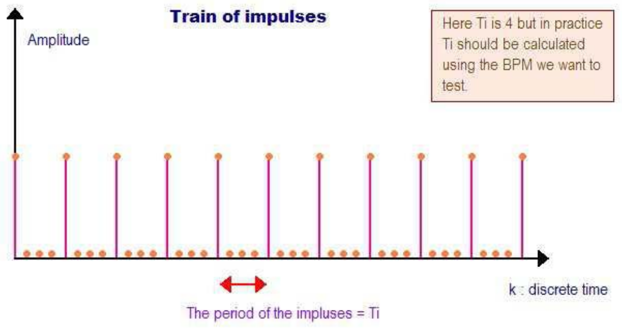
Train of impulses
Amplitude
Here Ti is 4 but in practice Ti should be calculated using the BPM we want to test.
k : discrete time
The period of the impluses = Ti

Figure 8. Pulse Train Illustration

However, due to poor performance of the previous algorithms, a library was used and implemented by the system for detecting the overall tempo. The result of the library resulted in each onset having individual tempo assigned to them, granted these tempos were similar having only minor offsets such as 84 BPM, 86 BPM, 80 BPM, etc. To utilize this, the average of all the tempos for each onset were calculated and rounded down, to be able to obtain the estimated tempo.

F. *Post-processing Module*

The post-processing module gathers all outputs of the three modules, pitch detection module, note segmentation module, and beat and tempo detection module, and transcribe these into tablature and sheet music format. This is done using the following steps:
1. Perform all feature extraction techniques, store the results in an array which includes Pitch, Chord Label, Chord Distinction, Onsets, and Durations.
2. Based on the tempo, calculate the actual note durations for each onset.
3. For each onset, transcribe the corresponding pitch if it is not a chord (based on the chord distinction array) and a chord label if it is a chord.
4. For transcription, the various pitch or chord label are then translated into the corresponding music elements for transcription into tablature and MusicXML.

The output for tablature is an ASCII file shown in a rich text box, which can be modified by the user. It can also be exported into a .txt file for saving.

The output for the music score is a MusicXML file, which contains the musical elements necessary to generate the music score. The MusicXML file is then imported into a third-party application to produce the music score file.

III. Results and Analysis

Testing was done on 22 audio files with varying elements. This includes 5 songs for basic testing, 3 songs that only contain chords, 6 songs that only contain single-notes, 4 songs that contain both single-notes and chords, and lastly, 4 songs that are excerpts from CD quality files (MP3 files which were translated to .WAV files). All songs with the exception of the 4 files which are excerpts from CD recordings were recorded through a microphone with a classical guitar. Six extra audio files with varying tempos are used for further testing the beat and tempo module.

Testing was done on the different modules namely pitch accuracy, onset detection accuracy, chord distinction accuracy, chord labelling accuracy, tempo accuracy.

Supporting data were also gathered from 4 testers. Two of the testers are professionals in the field and two are casual guitar hobbyists. They were given a survey after allowing them to use the system and test its capabilities.

A. *Pitch Accuracy*

The results for the pitch detection test are shown in Table I for a summary of the results. The testing was done only on the test data that contained single-notes. The aim of this test is to determine the accuracy of pitch assigned to the notes in the audio, based on octave errors, pitch errors, and both. Octave errors are defined as an assigned pitch that matches the pitch name but is in the wrong octave. On the other hand, pitch errors are complete errors in the assigned pitch. % Correct can have different basis, for example, '% Correct based on Octave Error' only consider octave errors in calculating the accuracy.

TABLE I. PITCH ACCURACY RESULTS

Category	Accuracy
% Correct based on Octave Error	92.92%
% Correct based on Pitch Error	76.79%
% Correct based on Both	69.71%

Overall, the accuracy is acceptable. It was observed that the auto-correlation algorithm mainly had problems dealing with the more complex audio files, such as those containing fast played notes, string echo, and those test data from the CD. It was also observed that majority of the error came from error regarding pitch, in which a completely different pitch from the expected pitch was given as the result.

B. *Onset Detection Accuracy*

The results for the onset detection test are shown in Table II for a summary of the results. The testing was done on the entire test data. The aim of this test is to determine the accuracy regarding detected note onsets, based on actual missed onsets and ghost note onsets. Missed onsets are onsets or notes that were played in the audio but were not detected by the system. Ghost notes are onsets detected by the system but were note played in the audio. For this module, two test cases were done, the difference being the second test case having a larger multiplier value (2 as compared to 1.5) for the running-mean algorithm. % Correct can have different basis, for example, '% Correct based on Missed Notes' only consider errors from missed notes in calculating the accuracy.

TABLE II. ONSET DETECTION ACCURACY RESULTS

Category	Accuracy
T1: % Correct based on missed notes	91.29%
T2: % Correct based on missed notes	74.25%
T1: % Correct based on ghost notes	53.45%
T2: % Correct based on ghost notes	94.97%
T1: % Correct based on both	44.75%
T2: % Correct based on both	59.09%

The results from the second test case were much more acceptable as compared to the results from the first test case. Upon observation it can be seen that ghost notes were the main reason of this poor accuracy. This is mainly due to the nature of the running-mean peak-picking algorithm. If note-onsets appear too slow, there will be a time in which small peaks which are generally just small background noise or static, will pass the threshold. This is shown in Figure 9.

Another analysis is that, in the case of test data containing many missed notes, this was mainly due to the speed at which these notes were played. It is possible to include these missed notes by varying the threshold, but this tends to become a problem because it will introduce the presence of even more ghost notes.

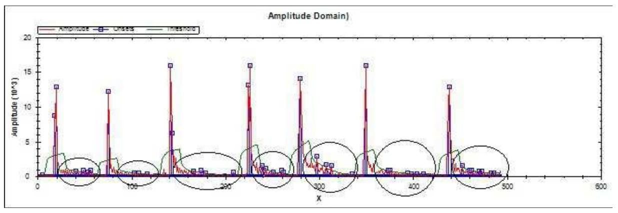

Figure 9. Ghost Notes Errors

In summary, although the second test run seems to overall have a better accuracy, the main reason for the low accuracy of the first test run are mainly due to the large amount of ghost note errors. Thus, if overall performance are more focused on individual data and detecting actual onsets, the first test run is considered to have achieved overall better results.

C. *Chord Detection Accuracy*

The results for the chord distinction test are shown in Table III for a summary of the results. The testing was done only on test data that contains chords. The aim of this test is to determine the accuracy of distinction between single-notes and chords. % Correct can have different basis, for example, '% Correct based on Single-Notes' only consider errors from single-notes which were incorrectly classified as chords in calculating the accuracy.

TABLE III. CHORD DISTINCTION RESULTS

Category	Accuracy
% Correct based on Wrong Single-Notes	95.03%
% Correct based on Wrong Chords	77.58%
% Correct based on both	86.14%

Overall, the results are generally good with over 85% accuracy. It can be observed that this more apparent in chords based on the results.

D. *Chord Labelling Accuracy*

The result for the chord labeling test is 85.04%. The testing was done only on the test data that contains chords. The aim of this test is to determine the accuracy of the chord labels assigned to chords detected by the system from the data. Chords that were covered for this research includes Major, Minor, Seventh, and Diminished chords.

It should be noted that the chords present in these test data consisted majority of Major and Minor chord structures, which are two of the basic chord structures. When tested for Seventh chords, the system was unable to differentiate it from its base chord structure (such as a Major Seventh chord being classified as a Major chord instead, or a Minor Seventh chord being classified as a Minor chord instead). Likewise, although it is able to distinguish Diminished chords, it does so with poor accuracy.

E. *Tempo Accuracy*

The result for the beat and tempo is 81.14%. The aim of this test is to estimate the accuracy between the estimated tempos of the data compared to its supposed actual tempo.

To get the supposed actual tempo, the following were done to assign a tempo to each test data:

• 120 BPM assigned to basic tests and nursery rhymes

• Information gathering on the internet for the tempos to the remaining test data

Extra testing was done using software generated audio files with pre-defined tempos which were then exported into .WAV files and presented as input to the system. The results can be seen in Table IV. As seen in the additional results, the accuracy is somewhat consistent, also being a bit over 80%.

TABLE IV. ADDITIONAL BEAT AND TEMPO RESULTS

Actual Tempo	Estimated Tempo	Accuracy
60	115	8.33%
80	75	93.75%
100	95	95%
120	115	95.83%
140	135	96.43%
160	160	100%
Average		81.56%

F. *Comparison with other Software*

Comparison was also made for some of the modules with some of the existing commercial software available on the internet, Melodyne [15[for pitch detection, Chordata [16] for Chord Labeling, and BPM Analyzer [17] for Beat and Tempo. A summary of the results are shown in Table V.

TABLE V. SOFTWARE COMPARISON RESULTS

Category	AGMT	Software Result	Software Used
Pitch Detection	69.71%	82.81%	Melodyne
Chord Labeling	85.04%	93.91%	Chordata
Beat and Tempo	81.14%	76.93%	BPM Analyzer

For chord distinction and note onset detection, there were issues encountered when testing or even finding software that deals with it. For chord distinction, no software that deals specifically with the problem was found. On the other hand, for note onset detection, there were software used for testing onset detection. However, there were issues mainly involving ghost notes. It was observed that, there were too many ghost notes detected, which may have come from the pick striking a string too loudly, effectively doubling note onsets detected for each actual onset. Due to this, results for note onset detection will also not be considered.

Melodyne was used for pitch detection. Results from using Melodyne were better compared to the system. It was observed that Melodyne also had some trouble dealing with those data sets that the system also had trouble with.

Chordata was used for chord labeling. Results from using Chordata were better and was also able to deal properly with Seventh chords and Diminished chords, as compared to the system. Additionally, Chordata also had a much more diverse chord structure database.

Lastly, BPM Analyzer was used for beat and tempo estimation. Results from using the BPM Analyzer were lower compared to the system. It should be noted however, that this is only a minor difference as tempo as discussed is subjective in nature. Thus, although the results are poorer, it is fairly negligible when perceived from human perception.

G. *Survey Results*

Generally, the testers were much more inclined to using guitar tablatures for playing guitar. It can be seen that the basis for this is that guitar tablatures are easier to read than musical scores. However, the testers agreed that guitar tablatures were lacking in some field such as consistency and application.

All the testers found the system useful. Generally, their evaluation of the system is that there can be further improvement made for the onset detection and pitch detection modules. They were happy however with the chord detection module as well as the inclusion of a beat and tempo (rhythm) for guitar tablatures.

As for the general efficiency of the output, they were satisfied with the resulting guitar tablature but had mixed opinions regarding the music score. A possible cause of this is that the music score generated by the resulting MusicXML was a bit confusing because of the missed notes and ghost notes, which made the resulting score seem a bit complex.

Additionally, the testers expressed an opinion of making similar software for commercial music. However, this is not in the scope of the research as working on this aspect requires being able to separate different instruments from a digital signal and working with different file formats (including compressed files such as MP3), which are different research topics altogether.

IV. Conclusion

Results of the research were generally acceptable. The problems were commonly found in the pitch detection and onset detection module. These problems occurred due to complex test data such as having fast notes played and string echoes. On the other hand, chord detection and beat and tempo module produced good acceptable results. The problems observed in these modules include confusion due to similar chord structures for the chord detection module and tempo comparison for the beat and tempo module.

The resulting tablature and music score generated by the system proves to be consistent with general tablature and music score structures. The only limitation regarding generating the music score is that MusicXML requires software to generate the corresponding music score.

The research was able to improve on the previous research by adding a chord detection and beat and tempo module, as well as being able to generate guitar tablatures.

References

[1] Macrae, R., & Dixon, S., “A guitar tablature score follower”, Multimedia and Expo (ICME), 2010 IEEE International Conference, Jul. 2010, pp. 725 -726.

[2] Kerdvibulvech, C., & Saito, H., “Vision-based detection of guitar players’ fingertips without markers”, Computer Graphics, Imaging and Visualisation 2007, CGIV ’07 , Aug. 2007, pp. 419 -428.

[3] Kerdvibulvech, C., & Saito, H. “In Markerless guitarist fingertip detection using a bayesian classifier and a template matching for supporting guitarists”, 2008.

[4] Tuohy, D., & Potter, W., “GA-based music arranging for guitar”, Evolutionary Computation, 2006, CEC 2006, IEEE Congress, 2006, pp. 1065 -1070.

[5] Franklin, D., & Chicharo, J., “Paganini-a music analysis and recognition program”, Signal processing and its applications, 1999, ISSPA ’99, Proceedings of the fifth international symposium, Vol. 1, 1999, pp. 107 -110.

[6] Kashino, K., & Murase, H., “Music recognition using note transition context”, Acoustics, Speech and Signal Processing, 1998, Proceedings of the 1998 IEEE International Conference, May 1998.

[7] Chua, E., Hernandez, R., Lautchang, W., & Ong, R., Guitar music transcription, 2009.

[8] Burk, P., Polansky, L., Repetto, D., Roberts, M., & Rockmore, D., Music and computers.

[9] Bello, J., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., & Sandler, “A tutorial on onset detection in music signals”, Speech and Audio Processing, IEEE Transactions, 13 (5), Sep 2005, pp. 1035 - 1047.

[10] Stark, A. M., & Plumbley, M. D., Real-time chord recognition for live performance.

[11] Davies, M. E. P., & Plumbley, M. D. “Beat tracking with a two state model”, International conference on acoustics, speech, and signal processing, 2005.

[12] Ellis, D. P. W., “Beat tracking by dynamic programming”, J. New Music Research, 2007, pp. 51–60.

[13] Patin, F., Beat detection algorithms, 2003.

[14] Alonso, M., David B., & Richard, G., Tempo and beat estimation of musical signals.

[15] Celemony, Melodyne, 2011.

[16] CLAM, Chordata, 2010.

[17] MixMeister, Bpm analyzer, 2010.