International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 6 – No. 4, September 2013 – www.ijais.org

A Novel System for Music Learning using Low Complexity Algorithms

Amr Hesham
Faculty of Informatics and Computer Science
The British University in Egypt
Sherouk City
Egypt

Ann Nosseir
Faculty of Informatics and Computer Science
The British University in Egypt
& Institute of National Planning
Egypt

Omar H. Karam
Faculty of Informatics and Computer Science
The British University in Egypt
& Ain Shams University
Egypt

ABSTRACT
This paper introduces a music learning system that uses new low complexity algorithms and aims to solve the four most common problems faced by self-learning beginner pianists: reading music sheets, playing fast tempo music pieces, verifying the key of a music piece, and finally evaluating their own performances. In order to achieve these aims, the system proposes a monophonic automatic music transcription system capable of detecting notes in the range from G2 to G6. It uses an autocorrelation algorithm along with a binary search based algorithm in order to map the detected frequencies of the individual notes of a musical piece to the nearest musical frequencies. To enable playing fast music, the system uses a MIDI player equipped with a virtual piano as well as section looping and speed manipulation functionalities to enable the user to start learning a musical piece slowly and build up speed. Furthermore, it applies the Krumhansl-Schmuckler key-finding algorithm along with the correlation algorithm to identify the key of a musical piece. A musical performance evaluation algorithm is also introduced which compares the original performance with that of the learner's producing a quantitative similarity measure between the two. The experimental evaluation shows that the system is capable of detecting notes in the range from G2 to G6 with an accuracy of 88.7% in addition to identifying the key of a musical piece with an accuracy of 97.1%.

Keywords
Music learning, automatic music transcription, key finding, monophonic music.

1. INTRODUCTION
Computer-based music teaching has been an ongoing field of research since the late 70's. There have been many attempts to automate completely the process of teaching to play Piano through techniques such as automatic music transcription, which is the process of inferring automatically the pitch, timing, and the duration of each played sound, given only the acoustic recording of a performance [1].

These techniques could not provide a comprehensive solution to overcome difficulties beginner self-learning pianists face while learning to play new musical pieces. The most important problem is reading music sheet music. Sheet music is a symbolic method of representing music to both performers and listeners, in the form of western music notation [2].

In addition, one has to know the name of a classical music piece in order to find its sheet music, which is not an easy task for beginners. Furthermore, common and interesting musically pieces are of a fast tempo. This prevents usually beginners from learning to play these pieces until they acquire skills and reach an advanced level.

Another difficulty self-learning beginner pianists face, when there is no sheet music available, is that they need to verify that the key of the musical piece is correct. A musical piece is "… an ordered collection of pitches in the whole- and half-step patterns" [3]. The key of a musical piece is found through analysis based on music theory; and it is the first degree of the scale of the musical piece. The key can be either a minor key or a major key.

After completely learning a musical piece, the self-learning beginner pianist needs someone experienced to listen to his or her performance and evaluate it. It is also necessary that the evaluation should be number-based, so that that self-learner would be able to see how much they have improved over time.

In this paper, a monophonic, i.e. only one note can be playing at any given time; piano music teaching system is proposed to help beginner pianists self-learn piano playing. The system helps eliminating the difficulties discussed above through offering a monophonic automatic music transcription module that uses the autocorrelation algorithm along with a proposed algorithm based on binary search to map the detected frequencies of the individual notes of a musical piece to the nearest musical frequencies. A fault detection and correction algorithm is also proposed in order to further enhance the accuracy of the autocorrelation algorithm while detecting note frequencies. The monophonic automatic music transcription module outputs its results to a MIDI file that is played on a virtual piano. The system also enables the learner to manipulate the speed of music without changing the pitch, and to loop arbitrary parts of the transcribed piece of music in order to tackle the difficult parts. Furthermore, the system is able to calculate the key of the musical piece using the Krumhansl-Schmuckler key-finding algorithm, which in turn uses the correlation algorithm. Finally, a new algorithm is proposed to evaluate the learner's performance of a musical

piece, compared to the original recording, in the form of a similarity percentage between the performance and the original recording.

2. LITERATURE REVIEW

The Literature review focuses mainly on previous attempts in the fields of computer-based music education and automatic music transcription. The field of computer-based music education is a vast field which focuses on enabling musicians to self-learn to play music, while the field of automatic music transcription focuses on converting a musical recording to a written format which can be easily read by musicians who desire to self-learn to play music using computer-based music teaching systems.

2.1 Computer-Based Music Education

According to Brandao et al. [4], there have been numerous attempts to use computers in music education. As a result, several systems have emerged that mainly try to accomplish one or a combination of the following tasks: Teaching music fundamentals, teaching musical performance skills, teaching music analysis skills and teaching music composition skills. These music education computer-based systems use techniques ranging from Computer-Assisted Instruction (CAI) to Intelligent Tutoring. CAI systems have a limited power since they do not have an internal representation of the user and therefore have minimal knowledge about the users. Consequently, they offer the same teaching experience to all their users since they cannot differentiate between the different users of the system. Intelligent Tutoring Systems (ITSs), on the other hand, maintain three types of knowledge [5]:

- Expert knowledge of the domain being taught, i.e. the system should thoroughly understand the subject being taught so that it can come up with solutions to potential problems.
- Student diagnostic knowledge, i.e. the system must understand the student's approach to learning in order to be able to correct any misconceptions.
- Curricular knowledge, i.e. the system must be able to reduce the difference between the student's knowledge and the expert's knowledge in order to facilitate the learning process.

2.2 Automatic Music Transcription

According to Scheirer [6], Automatic Music Transcription of audio data is the process of taking a series of samples from a sampled digital WAV file and transforming this low-level representation of data into a high-level representation such as standard notation, which can later be saved in the MIDI format. The music encoded in the digital data can be described in the form of three parameters: frequency, amplitude and the shape of the wave that represents the music [7]. Frequency correlates with the pitch of the sound, amplitude correlates with the loudness of the sound, while the shape of the wave correlates with the timbre of the instrument being played [8].

Costantini et al. [9] attempted to develop a polyphonic music transcription system for percussive pitched instruments. The main functions of the system were to capture note pitches and note onsets, which are the instants of note attacks. The Constant-Q transform was the signal processing technique that was used to detect note pitches by building a time-frequency representation of the signal. In order to detect note onsets, the short-time Fourier transform was operated on a frame-by-frame basis of the signal and an algorithm based on Support Vector Machine was used to identify the note pitch. The system had an accuracy of 97.3%. However, it needed to be trained by playing the whole note range of the used instrument. It also had to be trained on a number of songs by providing the songs and the corresponding music scores before the system could be used for the first time.

Benetos and Dixon [10] used a shift-invariant latent variable model for multiple fundamental frequency estimation and note tracking. The system had a good support of tuning changes and frequency modulations, such as vibrato, through incorporating shift-invariance and the constant-Q transform as a form of time-frequency representation, into the model. The system was trained on three variants: multiple-F0 estimation of orchestral instruments only, multiple-F0 estimation of orchestral instruments plus piano and multiple-F0 estimation of piano notes only. The system's accuracy was 57.9%, which means that almost half of the played notes would be misidentified.

An automatic piano tutoring system was proposed by Benetos et al. [11] which took as an input a recording of a student's performance, along with a reference score. The system synthesized a recording from the reference score and then transcribed both recordings using the non-negative matrix factorization algorithm. This method successfully eliminated transcription errors since they would appear equally in both recordings, and would therefore cancel out together. The system also used hidden Markov models for note tracking. The system compared both transcriptions and pointed out correctly played notes as well as mistakes on a piano roll presentation. The accuracy of the system was 93%. However, the musical score had to be provided as a reference, which defeats the purpose of automatic music transcription.

Guo and Tang [12] proposed a polyphonic automatic music transcription system that was able to detect computer-synthesized piano notes in the range of C3 to B8. The system functioned through converting the multiple-F0 estimation problem into a mathematical problem that was solved using mathematical treatment techniques including harmonic selection, matrix analysis and probability analysis methods. The algorithm reduced dimensions by initially applying principal component analysis (PCA) and then selecting candidates first by a human auditory model, and second by the harmonic structure of notes. The system had an accuracy of 80%. The computations, however, were complex and the system needed to be trained by playing the note range from C3 to B8 on a piano. The system also had to be trained on a number of songs by providing the songs and the corresponding music scores before it could be used for the first time.

3. PROPOSED SOLUTION

3.1 System Overview

The proposed system consists of four main components:

i) an automatic music transcription module which uses the autocorrelation algorithm along with two proposed algorithms for fault-detection and correction and frequency rounding to the nearest musical frequencies;

ii) a MIDI player module equipped with a virtual on-screen piano;

iii) a key-finder module based on the Krumhasnsl-Schmuckler key finding algorithm and the correlation algorithm, and

iv) a musical performance evaluation module which uses a proposed algorithm that compares two input signals and outputs a similarity measure for the two signals in the form of a percentage. Figure 1 shows the four main components of the system, along with the algorithms used to implement each one of them.

The system aims to solve the problem of not being able to figure out music by ear and having difficulties reading sheet music through offering a monophonic automatic music transcription module which receives a single-channel WAV file containing the music the user wants to learn, and converts it to a MIDI file which is played on the virtual on-screen piano.

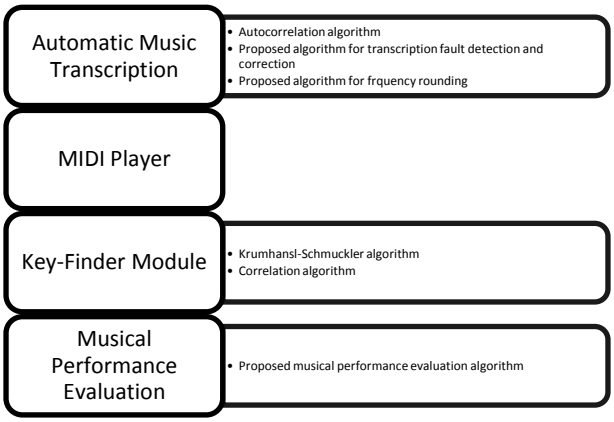

Figure 1: Main components of the system

The MIDI player module has two purposes. The first purpose is that it plays the MIDI file produced by the monophonic automatic music transcription module in order to help beginners to learn music playing even without having the ability to read sheet music. The piano highlights the key corresponding to the currently played note in a specific color (brown). It also displays the name of the note at the top so that the beginner can learn the notes corresponding to the different keys as they learn to play musical pieces. Figure 2 is a screenshot of the user interface of the virtual on-screen piano. The second purpose of the MIDI player module is to solve the problem of not being able to play fast music. This is done through offering two main functionalities: a speed trainer and a looper. The speed trainer enables the user to manipulate the speed of the song in the form of a percentage from the original speed. The looper functionality enables the user to choose any arbitrary part of the song and loop it any number of times. It also enables the user to loop at a variable speed.

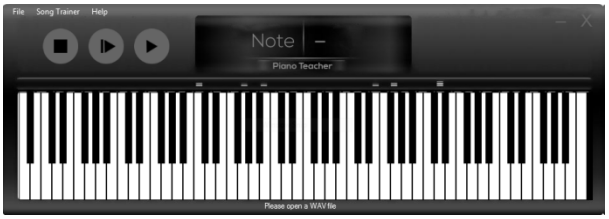

Figure 2: Screenshot of the Virtual on-screen piano

The key-finder module aims to address the problem of not being able to figure the key of a piece of music. It calculates the key of the musical piece and displays it on the top-right hand corner of the virtual piano. The user can then compare the key produced by the program to the key he figured out through analysing the chord progression of the musical piece. It is worth noting that the module calculates the most dominant key in case the song is based on multiple keys.

The last module, which is the musical performance evaluation module, aims to solve the problem of having to have a professional listen to the beginner's performance of a piece of music in order to evaluate it through enabling the user to feed into the system a single-channel WAV file containing the performance of the musical piece. The module then transcribes the performance version and compares its notes to the notes of the original version transcribed earlier. Finally the module outputs the evaluation in the form of a score in the range of 0 to 100, resembling the similarity between both versions of the musical piece.

3.2 Automatic Music Transcription

3.2.1 Architecture

Figure 3: Automatic Music Transcription module's architecture

The architecture of the automatic music transcription module, Figure 3, consists of five main elements. The first is signal segmentation, which divides the signal into a series of 20ms segments to be fed into the autocorrelation algorithm so that the fundamental frequency of each segment can be calculated. The next element is frequency rounding, which is carried out using a proposed algorithm based on binary search. The algorithm takes the detected frequency of each of the 20ms segments, and rounds it to the closest frequency, which belongs to a musical note.

The next element of the automatic music transcription module is the transcription fault detection and correction algorithm. Transcription faults, which are dealt with, are caused by the faulty estimation of the fundamental frequencies of each of the 20ms segments. The algorithm receives a list of rounded frequencies, which is the output of the frequency-rounding algorithm, and attempts to detect and correct as much as possible faults produced by the autocorrelation algorithm so that the transcription process would be more accurate.

The final step of the automatic music transcription process is converting the list of frequencies, which are the output of the proposed fault detection and correction algorithm, into MIDI notes by associating the key number of each note with its note on-set and off-set events and finally writing these MIDI notes to a MIDI file to be played on the virtual on-screen piano and to be saved to disk for later use.

3.2.2 Signal Segmentation

The first step of the automatic music transcription process is signal segmentation. In this step, the signal is divided into 20ms segments, which is equivalent to 882 samples of a WAV file sampled at 44,100Hz. This sampling rate corresponds to CD quality audio files and is therefore the

ideal sampling rate. The segments were chosen to be of a length of 20ms in particular as a result of extensive testing with different windows sizes. Larger window sizes make the system unable to detect short musical notes, while smaller windows sizes produce multiple identical frequencies, which are then grouped together by the transcription fault detection and correction algorithm, and thus requires more processing which hinders the system's performance.

3.2.3 Autocorrelation

From a signal processing point of view, the autocorrelation function is a correlation of a signal with a lagged version of itself. This enables the discovery of any hidden periodicity in noisy data [13]. The autocorrelation algorithm was chosen in particular due to its simplicity and accuracy compared to other frequency-domain pitch detection algorithms such as the FFT. The algorithm has a complexity of O(n2) and is calculated using the following equation [14]:

$$r_{xx}(\tau) = \sum_{n=0}^{m} x(n+\tau).x(n)$$

where τ is the lag and m is the number of samples on which the autocorrelation function will be applied. The autocorrelation function is repeated for each of the possible values of τ. It is worth noting that the window length should be at least twice the period of the longest note that should be detected. In case of a 20ms window, the piano note G2 is the lowest note that can be detected since it has a period of almost 10ms. The highest note that can be detected is G6 since the testing showed that autocorrelation algorithm is not accurate in case of frequencies higher than 1500Hz.

As can be seen from the equation, the autocorrelation function works through multiplying a signal with a lagged version of itself, and saving the product. The process is repeated for a range of lag values and then a graph of the autocorrelation function is plotted. Figure 4 shows a plot of 160 samples of a signal which will be processed by applying the autocorrelation algorithm. The resulting graph is that of figure 5. The period of the signal always corresponds to the second highest peak. A peak means that original and the lagged versions of the signal are similar. The higher the peak, the closer the match. Accordingly, there is always a peak at lag zero. This is because the two versions of the signal will be an exact match. The second highest peak results from applying a lag so that the two versions of the signal will be an exact match. This lag, which is almost equal to 38 in case of Figure 5, corresponds to the period of the signal.

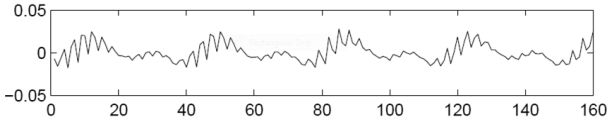

Figure 4: 160 Samples of a periodic signal. Horizontal axis represents sample number

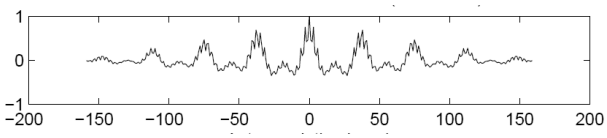

Figure 5: 160 Samples of a periodic signal. Horizontal axis represents lag value

In order to calculate the period of the signal, the following equation is applied:

$$T_p = K_p \times T_s$$

where T_p is the period, K_p is the lag corresponding to the second highest peak in the graph produced by the autocorrelation algorithm, and T_s is the inverse of the sampling rate of the signal. Finally, the pitch of the 20ms segment of the signal is the inverse of the period.

3.2.4 Frequency Rounding

Testing of the autocorrelation algorithm showed that it has an error rate of ±0.5% for identified pitches. This means that the produced note pitches cannot be converted directly into MIDI notes which are directly mapped to logarithmic musical frequencies. A simple solution to this problem could be a brute force algorithm of complexity O(n) which would compare each detected pitch to each and every musical frequency in a pre-specified range. This, however, is an inefficient solution that would consume excessive time to process a musical piece consisting of a large number of notes.

In order to solve this problem, an algorithm based on binary search is proposed to round each detected note pitch to the nearest musical frequency. The algorithm utilizes the power of binary search to divide the search space by half at every iteration and therefore has a complexity O(Log(n)). The algorithm starts by generating a list of musical frequencies in the range of the notes that can be detected by the autocorrelation algorithm. The algorithm then proceeds by rounding the frequency of each detected note pitch through setting the head pointer of the binary search algorithm to the index of the first musical frequency and the tail pointer to the index of the last musical frequency. The algorithm then keeps looping while the frequency is not yet rounded and the index of the head is not equal to the index of the tail.

At each iteration, the index is first calculated as the midpoint between the head and the tail and then the target frequency to be rounded is compared to the frequency at the index of the mid pointer. If both of them are equal, then the detected note pitch is already a musical frequency, and the algorithm moves on to round the next detected note pitch. Otherwise there is one of two cases: either the frequency at the index "mid" is smaller than the target frequency, or the frequency which has the index "mid" is larger than the target frequency. In the first case, one further step is taken to increase the efficiency of the algorithm;, and that is checking whether the target frequency is larger than the frequency at index mid, or is smaller than the frequency at index mid plus one. The target frequency is then rounded to either the frequency at mid or that at mid plus one according to which frequency is closer to the target frequency. The algorithm then moves on to rounding the detected pitch of the next note. This saves a number of iterations, and therefore makes the algorithm more efficient. Otherwise, the index of head is set to the index of mid plus one and the next iteration is started.

In case the frequency at index "mid" is smaller than the target frequency, then a check is made to determine if the target

frequency is smaller than the frequency at index mid, but is larger than the frequency at index mid minus one. In this case, the target frequency is rounded to either the frequency at mid or mid minus one according to which frequency is closer to the target frequency, and then the algorithm moves on to rounding the detected pitch of the next note. Otherwise, the index of tail is set to the index of mid-minus one and the next iteration is started.

The algorithm ends when all frequencies have been rounded. If, at any moment, head is equal to tail and the frequency at either of their indices is not equal to the target frequency, then the detected note pitch is out of the range of the note pitches that can be detected by the autocorrelation algorithm. This usually happens due to an octave error, which is a fault made by the autocorrelation algorithm in which a note is misidentified as being one or more octaves higher or lower than the correct note.

3.2.5 Fault Detection and Correction

In the autocorrelation algorithm step, some faults may occur while detecting the pitch of a given segment of a signal. The most common fault is the octave error, which means that the pitch of a note is misidentified as being one or more octaves higher or lower. Another common fault is misidentifying a note due to noise in the signal. This noise mainly results from recording music using non-professional hardware such as microphones and audio interfaces. A third problem which is very common, but is not due to a fault made by the autocorrelation algorithm, is that a note being played for more than 20ms will be identified as a number of notes each of a duration of 20ms. This produces an undesirable effect because each note will have a separate onset, and therefore the long note will sound discontinuous.

In order to attempt to solve these three problems, a new algorithm of complexity O(n) is proposed. The algorithm starts by creating two arrays of an equal length corresponding to the number of detected notes in the musical piece. The first array stores note frequencies, while the second stores the duration of each note in the form of time units, and therefore is initialized to all ones. Two variables, j and k, act as pointers to the first and last notes, respectively. The algorithm proceeds while j is not equal to k by checking if the current note length has exceeded the maximum allowed note length. In this case, no successive notes of identical frequencies can be merged with the current note. This ensures that a number of notes played separately won't end up being transcribed as a single sustained note. The algorithm then collects similar successive notes by comparing the current note to the next note. If they are identical, then they are combined into a single note that lasts for the duration of both notes combined. This ensures that a note that lasts for more than 20ms won't end up being transcribed as a series of 20ms notes with separate attacks.

The next step of the algorithm attempts to correct misidentified notes by comparing the current note to the note after the next one. If they are identical, yet the next note is different, then most probably it is misidentified and is therefore discarded and its duration is added to the duration of the first note. This operation is safe because at a note length of 20ms, it is difficult that there would be a note that lasts for such a very short time. The final step of the algorithm has to do with discarding notes that last for less than the minimum allowed note duration. In case such a note is found, it is discarded and its duration is added to the duration of the previous. Once again, this is safe since it is very unlikely that such an extremely short note can be played by a human being. Finally, if none of these conditions holds true, then j is incremented so that the next note can be processed.

3.2.6 MIDI Writing

In order to write the detected note pitches, after being processed, to a MIDI file, then each of the detected pitches is converted to a MIDI note number using the following formula [8]:

$$k_n = 49 + (12 \times \frac{\log \frac{f}{440}}{\log 2})$$

where "k_n" is the MIDI note key number and "f" is the fundamental frequency of the detected note. The MIDI key numbers are then written to a MIDI file, which will be played using the MIDI player.

3.3 MIDI Player

The purpose of the MIDI player module is to play the MIDI file that was produced by the automatic music transcription module. The MIDI player also enables the user to reduce the speed of the MIDI music to any fixed speed. It also allows the user to loop arbitrary sections of the MIDI file for an infinite number of times, giving him the ability to start at a stepped down speed and gradually increase the speed in an arbitrary number of steps.

3.4 Key Finding

The Krumhansl-Schmuckler key finding algorithm [15] was used to identify the key of the transcribed musical piece. It is based on the notion of key profiles, where each key profile consists of a vector representing the stability of each of the twelve pitch classes relative to that key profile. The key profiles were synthesized by performing experiments involving professional musicians by letting them listen to several musical pieces and having them decide how strong each pitch class is present in that musical piece. The values of the pitch classes are then linked to the key profile of the musical piece [15].

Figures 6 and 7 show Temperley's [15] revised key profiles for C major and C minor respectively. In order to get the key profiles for other major or minor keys, the values are shifted by the appropriate number of steps. For example, in order to get the key profile for D major, the C major pitch class would be shifted by two steps so that the D pitch class would have the value 6.35, the Eb pitch class would have the value 2.23 and the E pitch class would have the value 3.48. In case the G minor pitch class needs to be calculated, the C minor pitch class is shifted by seven steps so that the G pitch class would have the value 6.33, while the Ab pitch class would have the value 2.68 and the A pitch class would have the value 3.52.

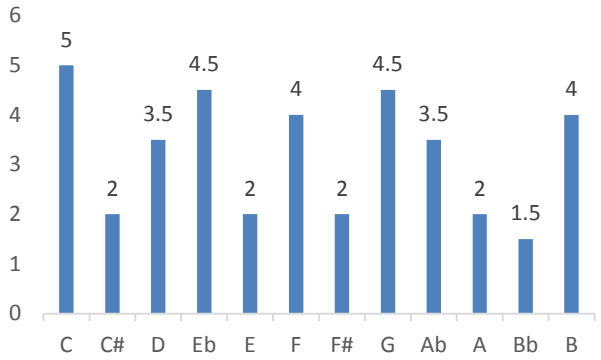

Figure 6: Revised C major key profile

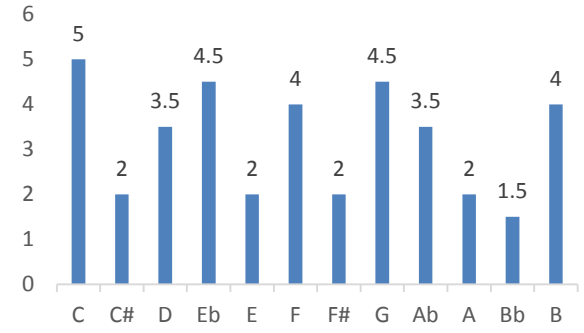

Figure 7: Revised C minor key profile

The Krumhansl-Schmuckler[15] key finding algorithm starts by generating the 24 major and minor key profiles form the C major and C minor key profiles. It then calculates the pitch class vector of the musical piece by counting the duration of each of the pitch classes of the musical piece. The algorithm finally correlates the pitch class vector of the musical piece with the other 24 major and minor pitch classes and chooses the key profile that has the highest correlation value to be the key of the musical piece, which is finally displayed in the label on the top-right corner of the graphical user interface.

3.5 Musical Performance Evaluation

The musical performance evaluation module enables the user to feed a recorded performance of the transcribed musical piece into the system, and to have the system evaluate that performance by outputting a score which is the similarity in percentage between the original version and the performance version of the transcribed musical piece. The performance version is also transcribed using the automatic transcription module for two reasons: the first is to identify its notes, and the other is to have common transcription errors suppressed as they will appear equally in both versions of the MIDI file.

The architecture of the musical performance evaluation module consists of a single proposed algorithm, of a complexity O(n). The algorithm starts by receiving two arrays of notes as input: the first one corresponds to the original version of the transcribed musical piece, while the second one corresponds to the transcribed performance version of the same musical piece. The two arrays are then labelled "reference" and "test" respectively if they are of equal length or if the first array has fewer elements than the second array. Otherwise, if the first array has more elements than the second array, then they are named "test" and "reference" respectively. The algorithm proceeds by first eliminating rests in both arrays and then counting the number of occurrences of each of the twelve musical notes in each array. An error measure is then produced as the sum of the absolute differences between the counts of notes corresponding to each of the twelve pitch classes in each of the two arrays. The algorithm then calculates the first similarity measure as the total number of notes in the test array minus the total number of errors and the difference is divided by the total number of notes in the original array. The quotient is then transformed to a percentage which is saved for later use. The equation below shows how the first similarity measure is calculated, where 'P' is the total number of identified notes in the performance version, 'E' is the total number of false positives and false negatives, and 'N' is the total number of identified notes in the original version.

$$S1 = \frac{P-E}{N} \times 100$$

The algorithm then calculates the second similarity measure by considering each note in the reference array and checking if there is an identical note in the test array at the same index. In this case, both notes are removed and the next note is processed. In case the current note in the reference arrays does not correspond to the note in the test array at the same index, but is identical to the note in the test array at the next index, then a false positive is detected. This means that an extra note has been played, and therefore two notes are removed from the test array, one note is removed from the reference array and a new error counter is incremented by one. In case the note in the reference array doesn't correspond to the note in the test array at the next index, but the note in the reference array at the next index is identical to the note in the test array at the next index, then a false negative is detected. This means that a note hasn't been played and therefore two notes are removed from the reference array, one note is removed from the test array and the error counter is incremented by one. In case none of these conditions holds true, then the note at reference can't be found in test and therefore the error counter is incremented by one and a note is removed from the start of both arrays. The process continues like this until any or both arrays are empty. If either of .the arrays ends up being not empty, then it has extra notes and therefore its count is added to the error counter.

The algorithm calculates the second similarity measure as the total number of notes in the test array minus the total number of errors and the difference is divided by the total number of notes in the original array. The quotient is then transformed to a percentage which is used to produce the score. This score is equal to zero percent if any of the two similarity measure is equal to zero percent, and is equal to the second similarity measure alone in case it is smaller than 30%. This ensures that a musical piece and its reverse won't be a perfect match. The number 30% was chosen in specific as a result of trial and error which showed that smaller numbers can make the algorithm less accurate in case the same notes are played in both versions but in different orders. It also showed that larger numbers can make the algorithm less accurate in case many errors appear between correctly played notes.

Finally, if none of the similarity measures are equal to zero and the second similarity measure is greater than 30, then the

score is calculated as the average of both similarity measures. This should increase the evaluation accuracy since faults made by one of the techniques used to calculate the similarity measure, will be corrected by the other. After the score is calculated, it is displayed in the label at the top right corner of the graphical user interface.

4. EXPERIMENTATION, RESULTS AND ANALYSIS

4.1 Automatic Music Transcription

The monophonic automatic music transcription module, which uses the autocorrelation algorithm along with the proposed frequency rounding and fault detection and correction algorithms, was tested on a dual core machine using six monophonic musical pieces. The six pieces were of different tempos and complexities, and were played on three different instruments: piano, guitar and saxophone in order to make sure that the transcription module is capable of transcribing sounds of different timbres. The guitar was used in specific to test the transcription module's ability to transcribe playing techniques such as vibrato and slides. Some of the musical pieces were recorded using a microphone, while others were recorded using direct line-in recording. This was done to make sure that the system is capable of transcribing music which has some noise due to being captured using a microphone.

In order to produce an accuracy measure, the following formula [11] was used:

$$Accuracy = \frac{N}{FP + FN + N}$$

where 'N' is the number of correctly identified notes, 'FP' is the number of false positives, which are notes that were identified by the system but were not played in the musical piece, and 'FN' is the number of false negatives, which are notes that were played in the musical piece but were not identified by the system. The accuracy was calculated as the average of the transcription accuracies of six monophonic musical pieces.

The accuracy of the monophonic automatic music transcription module was 88.7% when tested using the six aforementioned monophonic musical pieces. The module was capable of identifying notes played on any instrument in the range from G2 to G6. The average time the module takes to transcribe a five minutes monophonic musical piece was 18 seconds, which means that the module is capable of working in real time. Figure 8 shows the name and the accuracy of each of the six musical pieces. It can be seen that simple musical pieces, such as "Happy Birthday" have higher accuracies than more complex ones such as "Careless Whisper". It can also be seen that guitar music that was recorded clean, such as "Enta Omry" had a higher accuracy than faster guitar music that was recorded using distortion and multiple guitar effects, such as "Love Story" and the guitar solo.

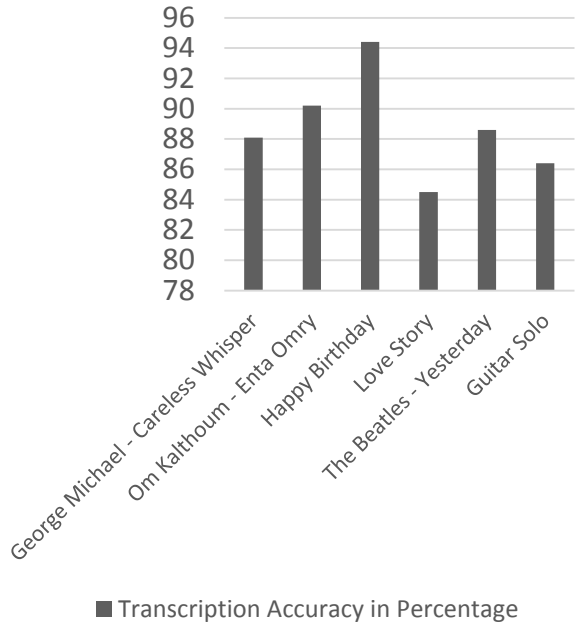

Figure 8: Transcription accuracy of the 6 musical pieces

The frequency rounding algorithm always produces correct results and has a complexity of O(log(n)) since it is based on binary search. The modification that was made to the algorithm, which enables it to peek at one position to the left or to the right of the middle at any iteration saves a considerable number of iterations that were wasted. This algorithm saves almost half of the time taken by the ad-hoc sequential search algorithm that would have been used otherwise.

The transcription fault detection and correction algorithm enhanced the accuracy of the monophonic automatic music transcription module by 9.2% on average. The algorithm mainly improves the transcription accuracy by combining similar notes and discarding very short notes that are not preceded or followed by identical notes. This gets rid of extra note onsets and also discards notes that don't belong to the original musical piece.

4.2 Key Finding

The Krumhansl-Schmuckler key finding algorithm and the correlation algorithm, together, were able to correctly identify the key of the six musical pieces since they were based on single keys. According to Temperley [15], the algorithm had an accuracy of 97.1% when tested on 48 preludes of Bach's Well-Tempered Clavier which is based on all 24 musical keys. This means that the keys of 44 out of 48 preludes were identified correctly, and therefore the algorithm is accurate enough to be used for the purpose of identifying the key of a musical piece so that a beginner pianist would be able to use this identified key as a reference while learning to analyze musical pieces.

4.3 Musical Performance Evaluation

The musical performance evaluation module was capable of producing accurate scores when comparing original and performance versions of each of the six musical pieces that were put to test. Although it is impossible to interpret what the

zero score means since signals can be completely different in infinitely many ways, the use of two similarity measures to calculate the final score enabled the production of a zero score in case the second similarity measure is less than 30% or in case any of the similarity measures is equal to 0%. This ensures that the original and the reverse versions of a musical piece won't score a 100%. This means that the algorithm can correctly produce a similarity measure between two signals, which can be used by beginners to seriously and individually evaluate their performances.

5. CONCLUSIONS

Technological advances in the fields of digital signal processing and computer-based music has made it possible to automate the music learning process to a large extent. This paper presented a solution to the four common problems faced by beginners trying to learn music on their own. The proposed system attempted to solve the problem of not being able to figure out music by ear as well as not being able to read music notation easily by introducing a monophonic automatic music transcription module which uses the autocorrelation algorithm along with two new algorithms to transform a monophonic WAV file to a MIDI file, which is played on the virtual piano. The system proposed to solve the problem of not being able to play fast music by enabling the user to manipulate the speed of the musical piece, and also to loop any arbitrary part of the musical piece and to variably manipulate its speed. The system was able to solve the problem of not being able to verify that a beginner-identified key of the transcribed musical piece is correct by introducing an automatic key-finder module using the Krumhansl-Schmuckler key-finding algorithm with the correlation algorithm. Finally, the system attempted to solve the problem of having to consult a musical teacher to evaluate a beginner's performance of the transcribed musical piece by proposing a new algorithm which is capable of evaluating the beginner's performance in the form of a score ranging from 0 to 100, resembling the similarity between the original and the performance versions of the transcribed musical piece.

REFERENCES

[1] Klapuri , A. & Virtanen , T., (2009), " Automatic music transcription," In Havelock, D., Kuwano, S., & Vorländer, M. Handbook of signal processing in acoustics, Springer New York, NY, Vol. IV, pp. 277-303.

[2] Good, M. (2001), "Musicxml: An internet-friendly format for sheet music", XML Conference and Expo, Montvale, NJ, February 2001, pp. 3-4.

[3] Scogin, N. (2010), Barron's AP music theory, Barron's Educational Series. N.Y.

[4] Brandao, M., Wiggins, G., & Pain, H. (1999) "Computers in music education," In Geraint Wiggins, AISB'99 Symposium on Musical Creativity, pp. 82-88.

[5] Burns, H. L., & Capps, C. G. (1988), Foundations of intelligent tutoring systems: An introduction, In Polson, M. C. & Richardson, J. J., Foundations of intelligent tutoring systems, Hillsdale: Lawrence Erlbaum Associates, pp. 1-19.

[6] Scheirer, E. D. (1995), "Extracting expressive performance information from recorded music", Master's thesis, Massachusetts Institute of Technology, Media Laboratory

[7] Moore, F. R. (1990), Elements of computer music, Prentice-Hall, Inc., NY

[8] Bello, J. P., Monti, G., & Sandler, M. (2000), "Techniques for automatic music transcription", In the First International Symposium on Music Information Retrieval ISMIR-00, Plymouth, Massachusetts, USA. October 2000, pp. 23-25.

[9] Costantini, G., Todisco, M., Perfetti, R., Basili, R., & Casali, D. (2010), "Memory Based Automatic Music Transcription System for Percussive Pitched Instruments", In *1st International Multi-Conference on Complexity, Informatics and Cybernetics (IMCIC)*, Orlando, Florida, USA, April 2010

[10] Benetos, E., & Dixon, S. (2011), "Multiple-F0 Estimation and Note Tracking for using a convolutive probabilistic model". Music Information Retrieval Evaluation eXchange, Miami, Florida, USA, October 2011.

[11] Benetos, E., Klapuri, A., & Dixon, S. (2012), "Score-informed transcription for automatic piano tutoring", IEEE Signal Processing Conference (EUSIPCO), Bucharest, August 2012, pp. 2153-2157

[12] Guo, Y., & Tang, J. (2012), "A Combined Mathematical Treatment for a Special Automatic Music Transcription System", Abstract and Applied Analysis, December, Vol. 2012, pp.13

[13] Hasegawa, B. H. (1987), "The physics of medical x-ray imaging. Medical Physics Pub Corp; June, ed. 2.

[14] Middleton, G. (2003). Pitch Detection Algorithms. Retrieved from the Connexions Web site: http://cnx.org/content/m11714/1.2/

[15] Temperley, D. (1999). "What's key for key? The Krumhansl-Schmuckler key-finding algorithm reconsidered". Music Perception, Vol. 17, pp. 65-100.