Miyaesi: Java Based Implementation for Automatic Music Transcription

H. Abeykoon, T. Kaushalya, N. Akram, A. Dissanayaka, S. Weerawarana, C. De Silva
Department of Computer Science & Engineering,
University of Moratuwa,
Moratuwa, Sri Lanka.

***Abstract*—Music Notes play a major role in the music world. They are extremely important for musicians and composers. Sometimes people like to know music notes of an already composed music which raises the need of music transcription. In the past decade up to now researches and engineers have come up with vivid techniques to do music transcription automatically using probability and signal processing. With the advent of computer science which facilitated encoding and recording music in digital format it became an important topic. Nevertheless, implementation specific details are still rare to find addressing Automatic Music Transcription (AMT). In this paper we discuss concepts behind automatic music transcription and how they are applied in the system Miyaesi, an automatic music transcription system implemented using Java programming language. Further, we discuss how time domain signal analysis and spectrum analysis leads to automatic instrument identification.**

Keywords-automatic music transcription, MIDI files, WAV Format, Windowing, Onset Detection

I. INTRODUCTION

In our research the primary target is providing a new algorithmic approach for mining out the musical notes of a monophonic music track which is in the .WAV file format leveraging the Java sound interface. Sub optimal goals of the implementation were generating a comprehensive music sheet with the decoded notes, converting music information into a MIDI file and allowing the user to edit the score and playback. The instruments being analyzed in the proposed system are limited to piano, violin, guitar and flute in the implementation, but the concepts can be extended.

Music transcription refers to extraction of musical notes and other related information from a given music piece. This is a difficult and challenging task done by skillful human musicians. Automatic Music Transcription is an algorithmic approach to simulate that cognitive process using signal processing techniques. The transcription task can be identified as the convertion of an acoustic signal into a MIDI like symbolic representation. In broad, music transcription is two folded as monophonic music transcription and polyphonic music transcription. Monophonic transcription refers to transcription of music data where only one instrument is being played at any given instance of time whereas in polyphonic it can be more than one.

Automatic music transcription had been researched for more than ten years. In many ways the problem is identified to be analogous to automatic speech recognition but has not received a sufficient academic or commercial value over time. Amazingly, speech recognition of a single person is not a completely solved problem. The main challenge in such a system is modeling human sound analysis using digital signal processing techniques and complex anthologies. Human brain is very good at estimations in real time. Unlike speech, music is always made out of discreet music segments which are triggered simultaneously. Sounds from various instruments are combined in various permutations and combinations which adds a dynamic complexity for music. Prediction of individual pieces and recognizing them individually is extremely hard even with existing complicated mathematics and statistics. Inherited noises existing in sound samples aggravate the situation. As an alternative solution, several websites that deliver the notes when searched from the title of a famous song are available. But that needs an online connection and the chance of not finding your song in their database is more probable.

Structured Audio Coding can be identified as an application of Automatic Music. It mainly focuses in arranging acoustic information into a compact format like MIDI. WAV file format is specified under RIFF (Resource Interchange File Format) specification to store sound data as digitized samples, whereas MIDI is a symbolic representation of music notes and events triggering at different times. Hence, converting a WAV file to MIDI falls into the problem of Automatic Music Transcription at the end.

A system which does the automatic music transcription should have several basic components which can be visible in any automatic music transcription system proposed in earlier researches as well. The difference is mainly the analysis methods, algorithms and interpretations. For example, frequency analysis is done applying a FFT (Fast Fourier Transformation), autocorrelation and zero crossing [1], [2]. Amplitude analysis with time is critical in an Automatic Music Transcription System. Many algorithms exist for smoothing waves, detecting onsets, detecting peaks etc. Most instrument classification algorithms have rule based music model built, and the matter of converting logics to probability is handled in various ways.

Implementation of such a system heavily depends on the APIs, Programming languages and libraries being used. Java is a very famous high level programming language from Oracle used by almost all in research, scientific and industrial programming tasks. It has rich APIs and libraries available which cooperate to build rich applications. Considering the above mentioned facts Miyaesi system was built using Java. Unlike traditional systems, our system differs from analysis perspectives and architectural perspectives. Several open source libraries were used along with the Sound Programming Interface provided with Java for the implementation. System specific details providing necessary mathematical information are discussed in brief in this paper.

This paper is organized as follows. In the second section which is immediately after introduction, General challenges in Automatic Music Transcription are addressed. Music file representation formats are introduced in the third section. Brief overview of related work on Automatic Music Transcription is provided in the fourth section. Fifth section describes Architecture of an enabling framework for AMT, while implementation approach is discussed in sixth and future research directions are explained in seventh.

II. CHALLENGES IN AUTOMATIC MUSIC TRANSCRIPTION

The challenge in music transcription is mainly the complexity of the problem. Even for a human musician to do an accurate transcription needs trainings of years. There is not a deterministic way how the music is composed, as thousands of combinations exist. Not only the note being played, but also the time duration each note is heard is different from each other. In polyphonic music when several instruments are played together extracting information that how many instrument are played in that instance, what are individual frequency information and amplitude information related to each instrument is extremely complex, and still an unsolved question.

Frequency estimation is usually done applying a FFT function over segmented data. A common problem arising there is harmonic sounds in the spectral structure. More over practical harmonics are not exactly an integral multiple of fundamental frequency. Finding the base frequency analyzing through the spectrum is one of the problems arising.

Inspection of a whole wave form of a music piece cannot be done with digitized data. We have to segment it down to individual parts for the sake of analysis. Then sizes of a segment, spectral leakages with rectangular windowing and correlations appearing between adjacent segments tend to cause problems.

To track time information of each note, boundaries of notes should be identified. This is usually done using amplitude variation of the acoustic wave form. But due to sudden variations, amplitude depending on the recording, noise added to the original signal creates a collection of problems which is sufficient enough to deceive a system implemented with simple rules and statistics.

Instrument classification is another research area combined with Automatic Music Transcription. Most of the techniques focus on music genre classification based on the features of extracted signals. Objective features like instrumentation, timbral texture, and rhythmic patterns are used for classification. Challenge in this process is identifying correct features that separate out similar instrumental sound together and different instrumental sounds to different groups. Features also differ depending on the technology used in the instrument – a violin can be a one with strings, metal wires or electric. The way of separating out time domain signal when several instruments are played together is extremely difficult.

Over time many researchers have tried to find answers to above problems, being partially successful.

III. MUSIC FILE FORMATS

Music can be stored in various file formats, some depending on the platforms and some not. In this section WAV, MIDI file formats are discussed in case they are related to Automatic Music Transcription (AMT).

WAV format w is a part of Microsoft's RIFF (Resource Interchange File Format) specification for the storage of multimedia files. A WAVE file is a type of RIFF file with a "WAVE" chunk specifying the data format and a Data chunk containing actual sound information. The default byte ordering of a WAVE file is little endian. If it is other way round that WAVE files are in RIFX format not in RIFF format. RIFF provides a way of storing all types of multimedia with different extensions. For example .RMI is for MIDI information while .RMN is for multimedia movie. In WAVE sound format data is not compressed, leaving out all signal information to be there making the file size relatively larger than a .mp3 file having same recording. Linear quantization of data is achieved using PCM (pulse code modulation). Data is stored with abstractions like frames and samples to be easy in the manipulation.

MIDI (Musical Instrument Digital Interface) defines a communication protocol between general computers and electronic music instruments. Earlier sequencers were built using hardware, but with high level languages like Java software based sequencers are available. In computers MIDI data can be streamed or sequenced. Standard MIDI files define a part of complete MIDI specification to address the timing issues that arise with MIDI wire Protocol. A standard MIDI file is a digital file containing only a set of MIDI events and time values to trigger them [13], [14]. Additionally, instrumental control information which is needed for playback is also included there. The standardization helps one software to create a MIDI file and different software or different type of computer can edit it and playback. All data values are stored in big-endian format with variable length. MIDI files are organized into data chunks (similar to RIFF files). Each chunk

is prefixed with an 8 byte header: 4 byte ID string used to identify the type of chunk followed by a 4 byte size which defines the chunk's length as number of bytes following this chunk's header. Header chunk contains chunk ID and size, format type, number of tracks and time division information. MIDI events contain MIDI Channel events, Note_on events, Note_off events, Note_After Touch events, Controller events, program change events and more.

Music Transcription system – Miyaesi uses WAV file format as the input. Formats like .mp3 is not suitable there because they use compression techniques to store data and also remove some original data preserving vital information. When all data is not present in a directly decodable format the error rate of the estimations during the analysis becomes high.

As WAV files represent digital encoding of acoustic sound and MIDI files represent a symbolic format in the high level converting a .WAV file to a .MIDI file is an application of Structured Audio Coding which comes under AMT. A true conversion between the two formats is unavailable as the problem of AMT is not solved yet.

Java sound programming interface is the API defined in Java to manipulate all types of sounds. It supplies mechanisms for manipulating system resources such as audio mixers, MIDI synthesizers, file readers and writers. There are other Java interfaces like JMF which is a higher level API for capturing and playing back time-based media as well. Javax.sound.sampled package has many methods and attributes defined to process sound data as objects.

IV. RELATED WORK

Different approaches to do music transcription exist. Following are some common researched areas which are related to AMT.

1. Frequency detection of input music or sound in general
2. Onset detection of notes
3. Instrument classification and identification
4. Temporal detection
5. Voice removal methods from a general song and retain musical part
6. Polyphonic music decoding to identify separate instruments

A. Frequency detection

Frequency detection is vital to identify the notes of a music peace because the notes are based on frequency. Signal processing has a close relationship with this area. Following are some of approaches which can be used to detect frequency in automatic music transcription.

1) Autocorrelation method

Autocorrelation is one of the approaches which work in the time domain which usually requires a number of periods of data from reliable estimate, and thus some averaging of the frequency signal is unavoidable. The method often exhibit difficulty in detecting the period of a periodic signal which is missing the fundamental harmonic in the harmonic series. [1]

2) Zero crossing method

Zero crossing is a method based on counting the number of times the amplitude crossed the zero amplitude level and scale the result into sampling frequency. Although this is a simple approach for pitch detection this does not work well with complex waveforms which are composed of multiple sine waves with differing periods. [2]

3) Fourier Analysis

Fourier analysis also has a few variations – Fast Fourier Transformation (FFT), Discrete Fourier Transformation, and Short Time Fourier Transformation (STFT).

In our project we have taken the approach of Fourier transformation in a specialized way for pitch detection. It is closer to STFT which is good for music transcription because it can give time information based on a moving window.

B. Onset Detection

Onset detection means detecting the beginning of the notes, using the amplitude level which is present. An onset detection function is a function whose peaks are intended to coincide with the times of note onsets. According to literature there are two main methods to detect onsets. First one is using amplitude or the power and the second one is using phase differences.

1. If an audio signal is observed in the time-frequency plane, the onset of a new sound has noticeable energy increase in the frequency bands in which the sound is not masked by other simultaneous components. Thus an increase in energy (or amplitude) within some frequency band(s) is a simple indicator of an onset.
2. When the phase of the signal is considered in various frequency bands, it is unlikely that the frequency components of the new sound are in phase with previous sounds, so irregularities in the phase of various frequency components can also indicate the presence of an onset. [3]

In our research we were more biased to the first solution in case it was in line with the STFT we have used.

C. Instrument classification and identification

Instrument classification can be mainly divided into three parts as

1. Identifying features that are different from one instrument to another.
2. Converting identified properties to numbers in order to feed them to a “Neural Network”.
3. Checking if the results are acceptably accurate.

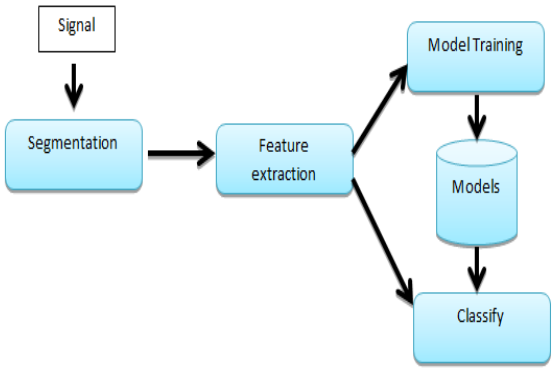

Figure1: General model for instrument classification

Time domain signal contains too much irrelevant data to use directly for classification. So feature extraction is inevitable and the results of classification will be purely based on the selection of correct features. Features can be further divided as temporal features and spectral features.

Spectral Features: Ceptral coefficients, Frequency and magnitude wrapping, Spectral centroid, Bandwidth**,** Spectral irregularities in harmonics (standard deviation of harmonic amplitudes from spectral envelope).

Temporal Features: Rising speed (average slope in the attacking phase), degree of Sustaining(length of the sustaining phase), Degree of vibration(sum of amplitudes of prominent ripples in the sustaining phase), releasing speed(average slope in the releasing phase)[4].

There are other important aspects when extracting features. Prior to analysis it is a must to normalize the amplitude values to a common range because the amplitude the instruments are heard can be changed from situation to situation. It is evident that one can select many features, but we have to select features, reduce dimensions, make them independent and go into the analysis. When classifying the instruments using feature vector, number of features we select, affect the accuracy of the classification. By using a sufficient number of features we can decrease the error rate and improve the classification results.

D. Temporal detection

In modern music tempo means "Beats per Minute". This feature has a relation with onset detection. A note can be one of three types - Crochet, Minim and Semibreve. A note duration is always an integer multiple of a beat. Thus in a music piece note duration is not same from note to note. But as a general rule beats per a minute is same for the whole music piece [5].After normalizing amplitude values and smoothing out the wave it is easy to detect the beats.

E. Voice removal methods

A WAV file has sound frames organized alternatively which are decoded and sent to left speaker and the right speaker. Voice removal methods take the advantage of this feature. In stereo, voice is placed centered but the instruments are not. Thus when the left channel and the right channels are separated out voice channel is presented in both channels – more or less with same intensity. So when the polarity of one channel is inverted and adds to the other voice gets cancelled out. But if the intensities are not exactly same in voice (it depends on the quality of the recording) some voice parts are retained in the resultant channel. This method is called "Center Pan Voice Removal" [6]. The advance voice removals are based on filters. Range pass filtering is generally used in such voice removing methods. Although advance techniques are used in various methods it is unable to remove voice with 100% accuracy.

F. Polyphonic music decoding to identify separate instruments

Separating simultaneously playing instruments from a polyphonic music is really complex. Many people have researched under this area and have come up with several high level probabilistic models.

Independent Subspace Analysis (ISA) method: This is similar to independent component analysis (ICA), but relaxes the constraint on the number of observed mixture signals. It can separate individual sources from a single–channel mixture by using sound spectra. The problem here is although this method is based on single independence in music signals there exist dependencies in both time and frequency domains. [7]

Convolutive Sparse Coding (CSC): This method eliminates the disadvantages in ISA method. Here the mixed signal is considered as a vector – a vector comprised of linear mixing of many source vectors. Weights are given to the sources by analysis. Both source vectors and the weights are assumed to be unknown. The sources are obtained by multiplying the observation matrix by an estimate of the un-mixing matrix. The main assumption in sparse coding techniques is that the sources are non-active most of the time, which means that the mixing matrix has to be sparse. The estimation can be done using a cost function that minimizes the reconstruction error and maximizes the sparseness of the mixing matrix. In this paper they have taken a 30-dimensional objective feature vector. They have used WEKA tool for classification [8].

Black Board System: In order to transcribe polyphonic music correctly it is needed to identify which frequency components are due to which notes. Each played note will typically generate frequency components at multiples of its fundamental frequency which are called harmonics. Moreover, notes which are harmonic to each other is actually played to make the music pleasing to human ear. Thus detecting such polyphonic notes are not easy at all. The blackboard system proper is a knowledge-based inference engine that can incorporate information from a variety of sources to produce hypotheses about (in our case) the notes present in the audio signal. The blackboard is a hypothesis database upon which initial input

observations are first written. There is also a set of expert agents, or knowledge sources (KSs) which are able to make inferences about some of the hypotheses that may appear on the blackboard. Each KS is an expert at doing some small part. When all the KSs get together and update the data on the blackboard at the end of the day a good result remains on the black board.

Genetic Algorithms for music transcription: This is a completely new approach to transcribe music. The underlying concept is as the polyphonic music creates a complex frequency lattice which is hard to decode. The new approach tries to reconstruct it using small pieces without trying to deconstruct it. Genetic algorithms are one of many tools used to explore large search spaces. The idea is to simulate the evolutionary process that species undergo in nature. Genetic algorithm simulates survival of the fittest, with the fittest individual at the end of our simulation being our solution in the search space. A collection of individuals are created in the Gene Pool. The less-fit individuals die away and the more-fit individuals live and go on to reproduce. Their reproduction creates slightly altered versions of themselves, using some sort of mutation of their genetic material. [9]

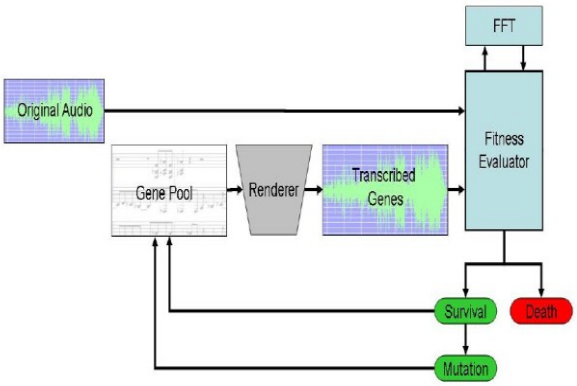

Figure 2: program flow of genetic algorithm. Extracted from [9]

Other than above mentioned methods, there are other methods such as Non-negative sparse coding, Non-negative matrix factorization (NMF)[10].
According to the researches we found that almost every technique mentioned above is very complex and implementation is time consuming.

V. MIYAESI SYSTEM ARCHITECTURE

Miyaesi automatic music transcription system is comprised of following components in the system implementation.

1. Load music file to the system
2. Preprocess the data for analysis
3. Estimate frequencies of segmented data
4. Onset Detection
5. Identify the instrument being played
6. Combine instrument information and frequency information to create a synthetic music (MIDI)
7. Generate a notation sheet
8. Facilitate editing notes and playback

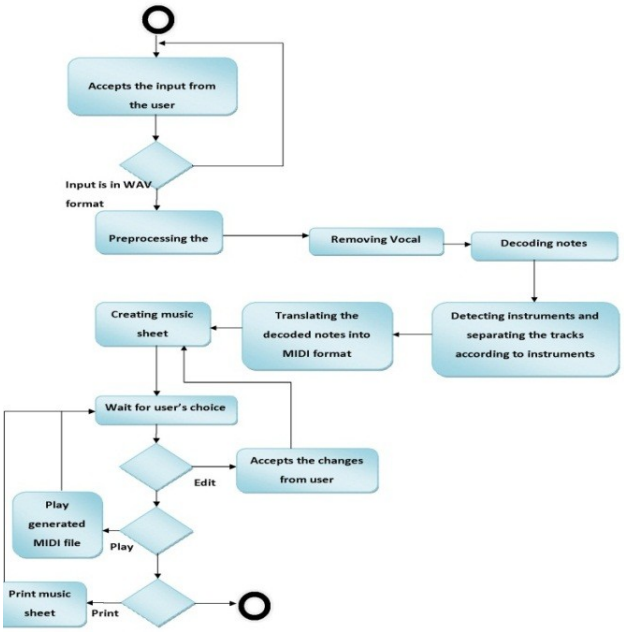

Figure 3: Activity Diagram of Miyaesi

Above mentioned components should interact with each other to do the transcription. There is a systematic flaw of data through the system. Most areas along the processing are sequential, because the output of one stage should be the input to the next stage. In order to gain the performance, owing to the above pattern of data flow the most suitable architecture is the "*Pipeline Architecture*". In the Miyaesi system this architecture is used to organize the above mentioned components to a logical order. The advantages of using pipelining are that any component on the way can be implemented separately as a class or package, independent updating is possible without affecting other tasks. To do experiments with a model that kind of environment is essential.

VI. MIYAESI SYSTEM IMPLEMENTATION

A. Loading Music information

The input format of the music information to the Miyaesi system is .WAV. To read a WAVE file Miyaesi uses Java Sound API, which facilitates reading a WAVE file to a pure byte array.
Samples are snapshots of an analog signal. During discreet time instances the values of the amplitude are captured successively and they are organized as a series.

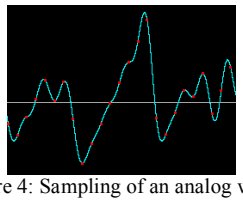

Figure 4: Sampling of an analog wave

After reading audio data as bytes following meta information can be extracted from the headers of the WAVE file.

- AudioFormat – If it is linear quantization (Pulse Code Modulation) this value is refered to 1. Values other than one means some compression is there.
- BitsPerSample – this value can be 8 or 16 representing the number of bits per sample.
- SampleRate – for most WAVE files this is 44100Hz.
- ByteRate (SampleRate*NumChannels*BitsPerSample) /8
- Number of channels – 1 means mono sound and 2 means stereo.
- Endian format – big or little endian

First an AudioInputStream is created. Then read the data in WAVE file is read as an audioInputStream to the application. AudioFormat can be extracted from the AudioInputStream. Some of the above details can be directly obtained from the AudioFormat and AudioInputStream, but some have to be calculated. For example

$$\boldsymbol{Sample\ Sound\ Length} = {\boldsymbol{Frame\ length}}/{\boldsymbol{Frame\ rate}} \quad (1)$$

$$\boldsymbol{Number\ of\ Equisident\ Points} = \boldsymbol{Sample\ Sound\ Length} * \boldsymbol{Sample\ rate/2} \quad (2)$$

The above information is vital for music transcription. Whatever analysis that has to be done should be performed on the byte array that resides in the computer memory after the WAVE file is read.

B. Preprocessing Data for Analysis

Before analyzing the digital data read from the file, they should get preprocessed due to several reasons – such as noise existence and format problems. Following is a brief discussion on a few common issues a sound analysis system faces.

1) Voice and Noise Removal of Stereo Data

Noise can appear in a WAV file due to a sound mixing problem at the studio or during format conversion. Generally noise has very high frequencies and random patterns. This makes identifying of noise part very hard. Professional systems ask user to input a noise sound profile (it has only the noise part recorded), and according to that estimates the noise of the rest of the sound track. With automatic transcription approach this is not acceptable anyway as it takes away the automation. General way is to set up a low frequency pass filter to the original and get rid of some noise (the audible range of human is from 20Hz to 20 kHz). Estimating the actual noise threshold is a problem as well.

If we are analyzing only the music information voice included in a WAVE file should obviously be removed. Then the same problem with noise arises. But thanks to the way a stereo wave file is recoded it is possible to remove some of the voice. In WAVE files frames are organized alternatively into a single byte stream such that odd samples go to the left channel and even samples go to the right channel. At the runtime channels are decoded separately and amplitude signals are provided to the left and right speakers. Using this organization of the frames we can separate out the frames representing left channel and the right channel. Then the polarity of one channel is inversed and the two channels are merged together again. If the voice was center-panned, that is voice components were equal on left and right parts they will cancel out because of the inversion. Using this method voice can be removed up to 80%, but not beyond. There are limitations in that method such as reverberation is preserved and instrumental information such as bass getting cancelled out as they are also center-panned. In a way it is good because bass sounds are not distorting out other important instrumental information.

2) Big endian-Little endian Conversion

The original bytes from the WAVE file itself do not represent the amplitude values. In our system we get two adjacent bytes combined to a short value. That is because generally in WAVE files one sample is represented by two bytes. If the file has been stored in little endian it is necessary to collect the bytes in reverse order before getting the short value. The sample count of the whole music sample is given by the following equation.

$$sample\ count = \frac{frameLength * Framesize * 8}{bits\ per\ sample * number\ of\ channels} \quad (3)$$

3) Segmentation

Segmentation of original data is another necessary thing that has to be performed before analysis. Most statistical methods tend to have loops. Hence time complexity will be high if the operations are applied on the whole data set. Further to apply FFT (Fast Fourier Transformation) segmentation of data with respect to some timing window must be done, because frequency information will tell only power of each frequency component. To assign the results to a particular time the data belong to that timing window only should be analyzed. FFT

functions usually require data points entered to be a power of two. In Miyaesi system a segment contains 16384 (2^{14}) bytes from the original byte array. (Hence converted to amplitude values it would be 8192).

4) Sampling frequency

In the preprocessing step we did not change the sampling rate because it destroys the matching between segments and respective time values. The sampling rate is usually 44100Hz, because according to Nyqiust-Shannon sampling theorem sampling frequency should at least be double as the highest original frequency (22 kHz).

5) Apply Windowing

An FFT applied on a data set produces spectral values riding on a curve, not a single point at a single frequency. Performing an FFT on values acquired during a finite sampling interval spreads out and distorts the results. If a rectangular window was used side lobes arise as a result of abrupt increase and decrease. Thus shaping the rectangular window reduces the spectral leakage. This is achieved by multiplying each endian converted value in a segment by the corresponding windowing value. Such standard windowing functions are Hanning window, Hamming window and Blackman-Harris window. The differences between these windowing functions depend on the degree of side-lobe roll-off. [11]

Figure 5: window shaping - Hanning Window

$$w_n Hanning = 0.5 - 0.5\cos(2\pi n/N) \quad (4)$$

$$w_n Hamming = 0.54 - 0.46cos(2\pi n/N) \quad (5)$$

$$w_n Blackman = a0 - a1cos(2\pi n/N) + a2cos(4\pi n/N) - a3cos(6\pi n/N) \quad (6)$$

Where a 0 = 0.355768, a1 = 0.487396,a2 = 0.144232, and a3 = 0.012604.

Miyaesi was programmed to allow the user to apply a windowing depending on the quality of the results he/she observes.

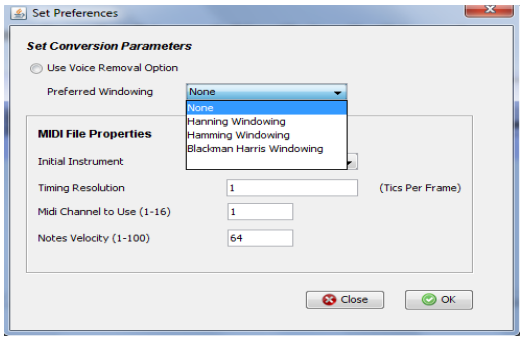

Figure 6: Options for windowing in Miyaesi

C. Frequency Estimation from Segmented Data

We used a STFT (Short Time Fourier Transformation) to convert time domain data to frequency domain. The original byte array was segmented to sub-arrays containing 16384 bytes (windowing size, which mapped into 0.09287 seconds) and endian conversion was done as described in previous section. There sub-arrays were the inputs to the STFT function we have implemented. Implementing a STFT is rather mathematical and standardized, so that for the implementation purpose it is very easy to find a standard algorithm written in Java.

The result from STFT is actually a two dimensional mapping or a spectrum between frequency bins and their respective power. The frequency represented by a bin (array position) is calculated using the following scale factor.

$$frequency\ scale\ factor = \frac{bin\ position * (sampling\ rate\ )/2}{number\ of\ total\ frequency\ bins} \quad (7)$$

The frequency f_0 related to the maximum power is chosen to represent the frequency of the wave of that whole segment analyzed. In other words, frequency of the wave at time segment t_t is f_0. As described, frequencies related to all segments are found. From preprocessing state time values related to segments were calculated and they were input to frequency estimating component.

The raw frequencies obtained from the above process do not always represent frequencies related to standard musical notes. Thus an approximation is needed. The simplest way to perform it is to compare the subtraction between the actual frequency and the adjacent standard note frequencies and make the approximation to the smaller gap.

Table 1: Mapping between a few notes and frequencies

NOTE	FREQUENCY (Hz)
C_3	130.81
$C^{\#}_3/D^{b}_3$	138.59
E_4	329.63

F_4	349.23

There are also other methods existing in the literature like auto-correlation and zero crossing [2],[11].

D. Onset Detection

An onset means beginning of a musical note. To create a note, duration of that note and the time value to trigger that note is essential. That task is performed by this component in the system. Input to the component is the whole array of endian converted values, and the output is a list of time values indicating where the onsets are observed.
Onset detection is an area which has been studied well. Various methods are proposed [3]. Two methods that were not very successful in the system implementation are discussed first. Prior to analyze amplitude values for onsets smoothing out the wave from is essential because amplitude variation of any music piece is very complex with spikes and sudden variations which hide the actual global patterns we want. We used following method to average out the wave form using following equation.

$$A_{segment} = \sum_{k=0}^{n} a_k / n \quad (8)$$

Where n is the number of discrete amplitude values present in a segment and *a* is the amplitude value. The season is during estimation of note edges it is not intended to go beyond the resolution of one segment.

1) Onset Detection Using a Moving Window

First method was based on a moving window. A window is maintained adding new values to the head and at the same time removing old values from the tail. Number of values in the window is kept as a constant. Another window with the same concept is kept in the front of the previous window's head as well. The mean values of two windows are compared at each new value. If the subtraction is significant consider it as an onset. The problem here is when the new value is passed from front window to the back window still the values are beyond the threshold when averages are compared.

2) Onset Detection Based on Increment Values

The second method was based on the increments between amplitude values. To be a candidate point on which an onset stands that point needs to be a local minimum. A simple check would be verifying whether adjacent values are greater than the one considered. If the increments at left and right of a candidate point are beyond some threshold it can be considered as an onset.

$$a'_1 = \frac{a_{t+1} - a_t}{(t+1) - t} \quad (9)$$

and

$$a'_2 = \frac{a_t - a_{t-1}}{(t) - (t-1)} \quad (10)$$

3) Miyaesi Onset Detection Technique

But in Miyaesiwe used a simple and straightforward way based on the normalization of data, mathematical concepts like confidence interval (CI), and acceptance and rejection of hypothesis. Next the first order differentiation of the averaged function was taken with respect to time. As it is done in a discrete manner it was simple enough to apply the following function.

$$\Delta f_k = a_{k+1} - a_k / t_{k+1} - t_k \quad (11)$$

Then each value was converted to the normalized value (considering mean and the standard deviation). This approach simplifies the information of music data to a standard normal function.

$$z_k = \Delta f_k - \overline{\Delta f_k} / \sigma_{\Delta f_k} \quad (12)$$

Data points having high modulus of the normal value (z value) can be considered as the points where amplitude of smoothed curve rising or falling relatively rapidly. Another thing to note is that as the time gaps between adjacent segments are equal, differentiated values are high where there is a high amplitude gap. Points with high amplitude variations relatively can be considered as onsets or beginning of true notes. This method gave best results out of the three methods discussed.

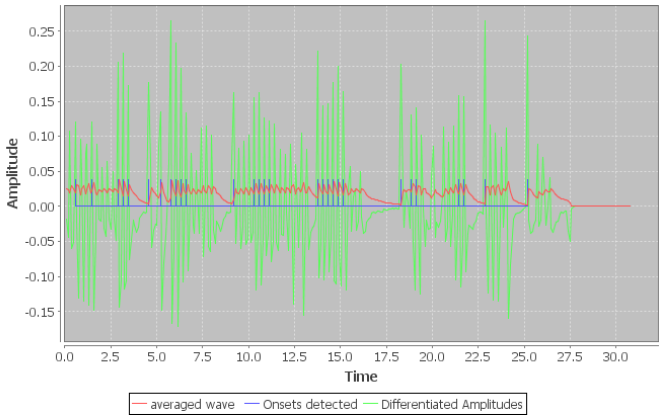

Figure 7: Smoothed wave, differentiation and onsets

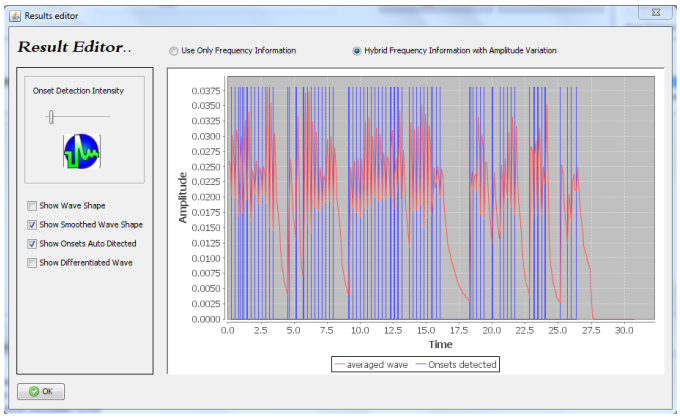

Figure 8: Onset detection and editing facilities available in Miyaesi

E. *Identify the instrument being played*

In order to perform a proper music transcription identifying notes is not enough. Musical instrument with which the note has to be played has a much illusion to hear the proper music. Even many available commercial applications which claim to do the transcription do not care about the instrument; they just allow the user to select a suitable one. In the system Miyaesi, we tried to classify features and identify the instrument with which the note between the particular onsets is played. As there is not a direct relationship with frequency and the instrument, we had to go for features based on temporal envelope of the wave shape and the spectrum relevant to that note. Six features were used to build the feature vector to train a neural network using supervised learning [4], [12].

1. Average gradient of rising phase
2. Average gradient of falling phase
3. Ratio between time of sustaining phase and the whole note duration period.
4. Ratios between areas under the waveform
5. Sum of moments of amplitudes about mid-point
6. Spectrum spikes count (in literature and by experiments we could find a connection between number of spikes in the spectrum and the music instrument)

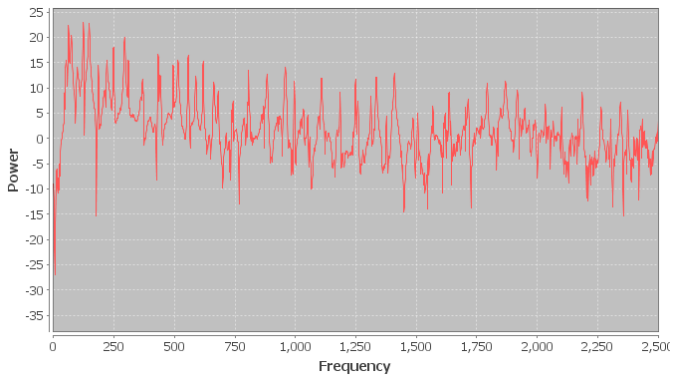

Figure 9: Power Spectrum for a note for a music sample

In Java "Encog" is a built library to facilitate neural network based trainings and classifications.

F. *Generation of MIDI files*

Following is a brief discussion on how MIDI data is organized and manipulated in Java.

1) *MIDI Sequence*

All information of a Standard MIDI file is referred to a sequence. A sequence contains one or more tracks. Sequences can be read from MIDI files or can be programmed from scratch. Also, a sequence can be directly written as a MIDI file to the system [14].

2) *MIDI Tracks*

Midi tracks contain the notes that are played by a single instrument. Software sequencer or a hardware sequencer can read the sequence and trigger the notes contained in each track at the correct time. Note On command starts a note, and Note Off command ends that note. A track can contain other events that do not correspond to notes but meta events. Track is a collection of MidiEvents [14].

3) *MIDI Events*

Midi event is a set of note information (Short Message) along with the time value to trigger the information. A short message contains note value (based on frequency), MIDI command (Note On, Note Off etc.), channel number to insert the event, and the velocity the note should be played. Except for short messages there can be SystemMessages and MetaMessages.

In Miyaesi, we input the approximated frequencies found at one of earlier stage, and the instrument information obtained from the previous stage into MIDI file generator component and got the MIDI file as the output, arranging the MIDI event appropriately. In memory MIDI file is represented by the sequence [14].

G. *Generation of Music Sheet*

In Miyaesi we used Java library called ABC4J to compose the music sheet from the notes being decoded. The input was a string representing the notes (according to its API) and as the output it generates a notation sheet which can be saved as an image or can be printed [15].

H. *Facilitate editing notes and playback*

This is necessary for an automatic music transcription system because transcription is not 100% accurate. The user of the system should be able to observe and correct any mismatch.

This can be easily done by coming up with a GUI to allow editing and reorganizing the MIDI messages and events.

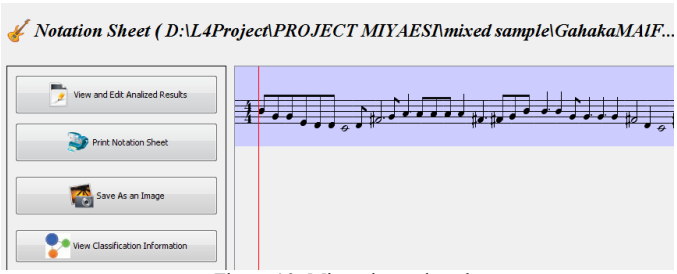

Figure 10: Miyaesi notation sheet

Playing a MIDI can be done directly using Java by playing out the sequence. If WAVE file is to be played it should be done by clips and buffers.

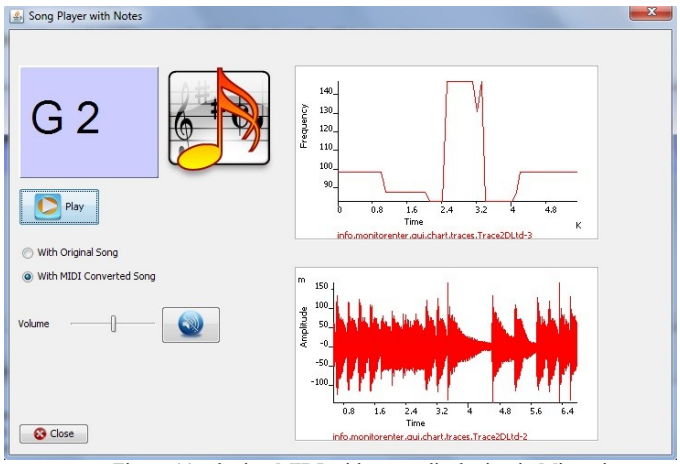

Figure 11: playing MIDI with notes displaying in Miyaesis

VII. RESULTS AND FUTURE RESEARCH DIRECTIONS

We performed two tests on the system. First one was to analyze WAV files having a tune with C D E F notes played sequentially with same time duration. Second one was to analyze WAV samples having one semibreve, 2 minims and four crochets from each instrument.

According to the results we obtained music instruments like piano, guitar and flute can be identified using above mentioned algorithms. Violin has too much amplitude variations caused by vibration of the instrument, making the onset detection to have false positives. This fact also makes same note values to appear several times rather than one. Another common fact that makes simple frequency decoding with a STFFT incorrect is that the harmonics present in instrument sound. It makes notes in higher or below octaves. Apart from the above short comings system works great delivering a notes sheet and a midi file generated out of a WAV file given.

The algorithm Miyaesi uses to find onset is composed of generic mathematics but its adaptation for AMT is novel. Combination of results from frequency decoding algorithm and onset detection algorithm has a large impact for the results. This shows that without going to complex analysis methods like Convolutive Space coding AMT can be achieved with acceptable accuracy.

Following figure gives a detailed analysis on correctness of instrument classification which is carried out with the system after it has been trained for 25 different synthetic music tracks from different instruments for different songs with different key signatures.

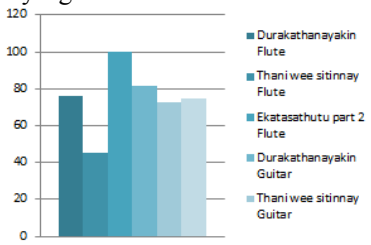

Figure 12: Correctness of instrument classification

This research paves way to many future research directions. A promising algorithm to filter out the base frequency out of a harmonic mixture will increase the accuracy of detecting correct notes. Smoothing of raw wave form to a low detailed form preserving important detail is needed in order to automate the music transcription. Further than that instrument classification can be enhanced using simple “rules” in music and paying attention to music patterns with a high granularity. Also during note detection incident details, pattern information is useful.

CONCLUSION

By nature the amplitude variation and the frequency variation of a music file has complex variations. Frequency variation can be fairly approximated by FFTs. For automatic music transcription analysis of amplitude variation can be done by applying Gaussian normalization to raw data. Accurate time matching is also required during above two analysis steps. Miyaesi Onset Detection Algorithm is a simple but robust algorithm to identify note boundaries compared to existing algorithms. For instrument classification Neural Network approach with statistical information withdrawn out of music recording gives good results if the design and the training of network are done with care. If the noise is presented with music data, above analysis framework would not work in the expected way. Further, for polyphonic music decoding much more complex analysis methods must be used because it needs to identify correlation between data related to the context.

REFERENCES

[1] Patricio de la Cuadra, Aaron Master_ and Craig Sapp,"Efficient Pitch Detection Techniques for Interactive Music", publications of *Center for Computer Research in Music and Acoustics, Stanford University*

[2] "Pitch detection algorithm" [Online] Available: http://en.wikipedia.org/wiki/Pitch_detection_algorithm [Accessed: March 21, 2011]

[3] Simon Dixon, "Onset detection revisited", In Proceedings of *the 9th International Conference on Digital Audio Effects (DAFx-06), Montreal, Canada*, 2006.

[4] Tong Zhang , "Instrument Classification in Polyphonic Music Based on Timbre Analysis" , Proceedings of *Hewlett-Packard Laboratories*

[5] "Tempo, Wikipedia" [Online] Available: http://en.wikipedia.org/wiki/Tempo [Accessed: February 12, 2011]

[6] "The Truth About Vocal Eliminators" [Online] Available: http://www.ethanwiner.com/novocals.html [Accessed: February 23, 2011]

[7] M. Casey and A. Westner, "Separation of Mixed Audio Sources by Independent SubspaceAnalysis", in Proceedings of the *International Computer Music Conference, ICMA, Berlin*,August, 2000.

[8] P.S. Lampropoulou, A.S. Lampropoulos and G.A. Tsihrintzis, "Musical Genre Classification of Audio Data Using Source Seperation Techniques", *Department of Informatics, University of Piraeus*.

[9] David Lu!!, "Automatic Music Transcription Using Genetic Algorithms and Electronic Synthesis ", April 25, 2006.

[10] Nancy Bertin, Roland Badeau, Ga¨el Richard, "Blind Signal Decomposition for Automatic Transcription of Polyphonic Music: NMF and K-SVD on the Benchmark", *Signaland Image Processing Department, GET-Telecom Paris*.

[11] Richod Lions,"Windowing functions improve FFT results" [Online] Available: http://www.tmworld.com/article/322450-Windowing_Functions_Improve_FFT_Results_Part_I.php. [Accessed: May 2, 2011]

[12] Wenxin Jiang, Alicja Wieczorkowska, and Zbigniew W. Ras, "Music Instrument Estimation in Polyphonic Sound Based on Short-Term Spectrum Match", publication in *University of North Carolina, Department of Computer Science, Charlotte*,2007.

[13] "Java Sound API"[Online] Available: http://java.sun.com/products/java-media/sound/reference/api/index.html [Accessed: May 3, 2011]

[14] "Understanding and Using Java MIDI Audio" [Online] Available: http://www.ibm.com/developerworks/library/it/it-0801art38/ [Accessed: February 10, 2011]

[15] "ABC4J" [Online] Available: http://code.google.com/p/abc4j/ [Accessed: February