Note Recognizer: Web Application that Assists Music Learning by Detecting and Processing Musical Characteristics from Audio Files or Microphone in Real-Time

Markos Fragkopoulos, Athanasios G. Malamos(✉), and Spyros Panagiotakis

Media, Networks and Communications Lab, Department of Informatics Engineering, P.O. Box 71500, Heraklion, Crete, Greece
exog3n@gmail.com, {amalamos,spanag}@ie.teicrete.gr

Abstract. Note recognizer is an online web application. In order to overcome the performance issues of the internet infrastructure (browser, devices, OS platforms) traditional algorithms have been re-designed and novel processes based on the Web Audio API have been implemented. It is the first time that open standard web tools offered in all the commercial browsers are used to build an application that usually required dedicated signal processing libraries. These novel processes and algorithms provide MIDI (Musical Instrument Digital Interface) information out of audio files or microphone. Our application may assist musical education by allowing students to transform their inspiration or a performance into notes.

Keywords: Web audio · Javascript · HTML5 · Pitch detection
Onset detection · MIDI

1 Introduction

Automatic Music Transcription (AMT) is the conversion of an acoustic signal (music) into a formal musical representation, such as a MIDI file or a score format. It is considered to be a significant technology area in music signal processing. In [10] it is defined as the process of converting an audio signal (natural or recorded) into a piano-roll notation (a representation of musical notes across time), while in [11] it is defined as the process of converting a music signal into a common music score. A typical formal music representation format includes pitches, onsets and offsets of the notes, tempo and (ideally) the instruments [4].

Automatic music transcription (AMT) could be analyzed into the following major tasks:

– Pitch detection, which is the process to estimate the pitch or fundamental frequency of a digitized music sample [4, 5]. This process results to the music note or tone that corresponds to the sample. This process may be performed in time, frequency or both domains according to the algorithm we choose

A. L. Brooks et al. (Eds.): ArtsIT 2017/DLI 2017, LNICST 229, pp. 403–412, 2018.
https://doi.org/10.1007/978-3-319-76908-0_39

- Note onset detection that is the beginning of a musical note [4, 6]. It is affected by the transient of a note switch but it is not expressing this transition which is independent by the note but rather the beginning of the pure note waveform. Both pitch and onset detection are considered open problems.
- Tempo estimation, which refers to the perceived tempo of a music sample [4]. It is an objective estimation that differs even between human experts that hear the same song. It is considered as an open problem as well, thus new algorithms appear every year in the competitions (ex MIREX).

In the AMT tasks we may include also some pre-conditioning actions that may be applied to the audio signal that improve the music characteristics estimation efficiency such as, Loudness Estimation that is related to potential amplification of the signal and instrument related filtering that might improve the pitch estimation.

In the past years, the problem of automatic music transcription has gained considerable research interest due to novel high impact and widespread applications, such as automatic search and retrieval of musical information features, interactive music systems (e.g. computerized performances, score following, and rhythm tracking), as well as musicological analysis [11, 12, 14]. The current research efforts focus in the development of innovative algorithms that extract efficiently musical features of multi-instruments recordings [4]. The problem of single instrument AMT may be considered solved from the algorithmic and fundamental research point of view. On the other hand, production and usability of a service or a product is not only a matter of algorithmic efficiency but furthermore is a matter of viability of production under the customer and market requirements. In the case of music applications, the new area of interest for customers and professionals is considered to be the web [8, 9]. The web improves accessibility and makes easier the application adoption and success. However, a significant disadvantage of the web is that applications are executed in the user (usually portable) device and we have to be sure that even a "weak" device may "decently" execute the application. Thus, web requires applications to be of low computational complexity.

In the market there is a limited number of applications that extract music information from recordings and even less applications that may be able to extract information in real-time (e.g., Ableton Live 9, AudioScore, ScoreCloud, AnthemScore). There are also some key demos over the web like Pitchdetector and Beats Audio API. Especially in the case of web or mobile platforms, there are no efficient applications appeared.

The idea for this application came from the music school student's involvement with music in all its aspects (electronic/instrumental). The main motivation for the creation of this application is the need of any student musician to log those scattered moments of inspiration and creativity and maintain them documented as ideas. Therefore, knowing that inspiration is coming suddenly, any minute, any place, this application has to overcome the barriers of a standalone application (like those appearing in the market) that requires personal computers to execute. Instead the purpose of this work is the development of a fully web-based real-time application that can be executed seamlessly in a mobile phone or tablet or a personal computer, free of any application installations and plug-ins. As might be expected, web increases

significantly the usability and usefulness of the application, however, imposes additional constraints to the, anyway, difficult problem of Automatic Music Transcription.

The implementation of this application is presented over the next chapters of the paper. The evaluation that is quoted has been done with some different instrument samples in two specific axes: note-by-note separately and comparative to some other similar commercial software.

2 Implementation

Note recognizer was developed as a proof of concept of an online automatic transcriber. In this application the user can track the recording of music notation over time on piano roll. The minimum requirements of the system limited only on a computer with satisfying performance capabilities, a major browser and a microphone or a music file. The user also can set a few details for more accurate results. To accomplish the purposes of this work, the implementation needs to interlock a series of functionalities. These consist of audio sound routing and processing, onset detection, pitch detection, tempo calculation, characterization of music events, note representation and interoperability mechanism. This application implements a part of the algorithms studied and creates some more. It was decided to present a limited range of functionalities, that the application is first stable, functional and give its mark on future objectives. At the end of this paper we will discuss some of the candidate features for future Note Recognizer features.

2.1 Sound Routing

The application sound routing and retrieval achieved with an implementation using the Web Audio API along with custom algorithms. Instead of using PCM data for retrieving the audio signal in our workflow, AnalyserNode (Web audio API) have been engaged. The AnalyserNode interface represents a node able to provide real-time frequency and time-domain analysis information. It is an AudioNode that passes the audio stream unchanged from the input to the output, but allows you to take the generated data, process it, and create audio processing functions.

2.2 Music Information Retrieval

In music information retrieval there are three basic estimation procedures: onset detection, BPM calculation and pitch detection.

Onset detection and BPM calculation. Onset detection, typically follows a flow of actions in order to increase efficiency [1, 7]. Initially, pre-processing of the raw audio signal to increase the performance of the following steps, generation of the Detection Function, post-processing, applying a peak selection algorithm in the detection function to select adequate peaks (Fig. 1).

Pre-processing indicates the transformation of the original signal to the weakening or strengthening of some aspects of the signal. For the pre-processing step

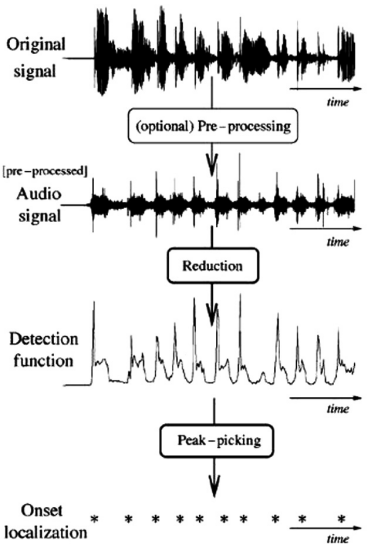


Fig. 1. Flow chart of a typical onset detection algorithm [1]

implementation of Short-Time Fourier Transform (STFT) have been engaged [1]. The Detection Function that is implemented is the High Frequency Content.

$$f(x) = \sum_{m=-\frac{N}{2}}^{\frac{N}{2}-1} x(\mathrm{ns}+\mathrm{m})H(m)e^{\frac{2\pi mk}{N}}$$

where n is the time, s is the window step size, i.e., H (m) is the Hamming windowing function, which reduces spectral smudging and the k frequency. When using the STFT, we do not use a sliding window, instead we hop window by step s to create successive STFT overlapping Windows.

For the implementation a Dynamic Compressor (Web audio API) have been used for pre-processing the sound. The *DynamicsCompressorNode* interface provides a compression effect, which lowers the volume of the loudest parts of the signal in order to avoid clipping and distortion that may occur. In general, a clearer sound without distortion may improve onset detection algorithms accuracy and performance.

The process is as follows: An audio file or microphone is loaded in webaudio component. If it has multiple channels they are summed into a single channel. Then the music signal is cut into overlapping sections. A windowing function is applied; in this case, the algorithm uses a hamming window. Apply the FFT algorithm with some zero padding to the windowed, overlapping signal. Then move the next frame of music and repeat this procedure until the recording has stopped and all parts of the signal have been processed. While the general procedure of the STFT is straightforward, the STFT has two tunable parameters that need to be appropriately adjusted: the percentage of overlap between successive frames and the time length of each window.

The windowing function is used to reduce smudging of adjacent frequency components. This is done by multiplying the time domain signal by the windowing function. Using a windowing function comes at a cost. The windowing function will decimate the amplitude of the signal, throwing away information. To solve this, each STFT window has some percentage of overlap between its current window and its previous windows so that if an onset is located in a weak part of one window, it is caught by the next window; unless otherwise stated, an overlap of 50% is used in all cases.

High Frequency Content and Post-processing. HFC based on the assumption on which: onsets have most high-density energy in areas where the mixing with other parallel components is lower, a condition that usually occurs in high frequency regions. This can be utilized by weighting each STFT window with a factor proportional of its frequency [1].

Post-processing used to facilitate procedures of thresholding and peaks selection by increasing the uniformity and consistency of the events related to the characteristics of the detection function, ideally turning on isolated readily detectable local maxima. Normalization of the signal is described in the theoretical implementation as part of the STFT which is used in pre-processing. In current application normalization have been implemented after the detection function because the quality of the results was better in this position after various tests. Then a moving median filter [2] is applied based on a previous implementation in [17], which is customized to run on the client-side. Afterwards the only remaining task to get the onsets is a dynamic threshold picking method which is built from scratch.

For the BPM calculation some code snippets from Beats Audio API have been implemented including the counting of intervals between nearby onsets, where is measuring time duration between beats [3] and grouping the neighbors by tempo where is produced a statistical histogram of measurements. Then by extracting the most significant (according rules) periodicity a tempo estimation, in beats per minute (BPM), can be achieved [13].

Pitch detection. Pitch specified as property of an acoustic signal which is determined by the frequency of the waves that produce it, so the pitch represents how high or low the sound is. For music, the assessment pitch corresponds the detected frequencies in some musical notes. For more accurate results we need to cut some amount of noise form the signal before the pitch detection algorithm. This is achieved by implementing a parametric EQ with some instrument oriented presets. These presets based on each instrument frequency range and can be selected by the user from a simple select box. According to the pitch detection code snippets from Wilson C. PitchDetect demo [15] have been implemented. So the method that been used for pitch detection is Auto-correlation. This is the cross-correlation of a signal with itself at different time points, in other words is the similarity between observations as a function of time elapsed between them [16].

2.3 Music Information Processing

The main requirements posed by this application during the information processing are the results pump ability from the algorithms in real-time and the handling of them in time oriented way.

Thread and snapshot handling. To achieve extraction results by the algorithms in real time there should be a continuous flow of the incoming signal to these algorithms, and in the same time a continuous recording of the music information by the algorithms in the system. For these purposes the application uses an execution thread (thread) as a key pillar implemented with the *setInterval*() javascript function. This thread basically sends and receives information from subroutines finite times (cycles) per second. From each cycle a single snapshot is produced whose time position is equal to the deviation of the time position of the current cycle from the time position of the first cycle. The flow of the incoming signal to the detection and calculation algorithms controlled by two AnalyserNodes. The allocation of the detection algorithms results made in a central snapshots array, whose positions are directly-linked with cycles and thus with time. We can say that as a digital audio signal is expressed in an array of values, so the music information of this signal is expressed by its snapshots array. The sampling rate of the snapshots array is different from the signal sample rate and equal to the number of cycles per second that thread executed. In the present application the sampling rate of the snapshots array is about 50 Hz.

Auto-correction. Through constant experimentation we estimated the accuracy of the autocorrelation algorithm. Errors usually have to do with noise and reverberation. Improving the algorithm itself was not possible due to lack of advanced theoretical knowledge in signal processing. Hence, we decided to use some reasonable assumptions to achieve rudimentary correction of the results in real-time. In fact, the main tool in this case is the snapshots array through which we can make time based comparisons between snapshots and apply some basic rules. Thus, in each cycle the executing thread calls a function that examines music information snapshots of the present note along with the previous and following two notes.

Characterization and objects. The characterization is based on the onset and the offset of each note with the logical assumption that each onset of a note that detected has also an offset and all intermediate snapshots have the same pitch. This function is based on the information of the offset of each note so it cannot operate in real-time as if one note is played at a particular time the system does not know in advance when it will end. So this algorithm is executed serially in snapshots array with a small delay. The results are recorded in the system, but only on demand of the user are represented and shall replace the previous ones. More specifically, when the algorithm detects that a snapshot is an onset and another is an offset of a certain note then finds the most common pitch between snapshots and defines the pitch of the note. Then is the moment that the note object which consists of snapshots created. By characterizing the items, we can correct pitches and instances for which we have no information. The auto-correlation algorithm recognizes them as gaps or errors, providing they are within the limits of an onset and offset. Each notes' offset, detected by a function which sets a

threshold and in a particular sequence comparisons of the wavelength of each snapshot, provides a snapshot as the end of the note.

2.4 Music Information Representation

Time and space relevance. The representation of the diluted music information based on the conversion of time in space. The key factor here is the time and space relevance. It would be very simple if the display of musicological characteristics took place in demand, this because we would have the total length of the signal and we could easily compute the relativity of space and time. Seconds are not the only unit of time to be used in the application, also used musical units beat and bars which duration, such as their length on axis x, determined by the tempo. When the render of the music information should be achieved in real-time we do not know the duration of the audio clip. We also know that the algorithm for calculating the tempo make some initial time to calculate the first result. These two parameters are indispensable for fixing sizes in space. To overcome this problem, the application employs two initial assumptions. The first is that the piano roll (canvas) has a finite size, and the second that the application starts with an agreed average of 120 BPM, until the tempo calculation algorithm return the first result. So the sizes of the measures planned at the start of the application base of calculations based on the original values and then by user request can be redesigned based on new figures calculated based on the actual speed of the music clip.

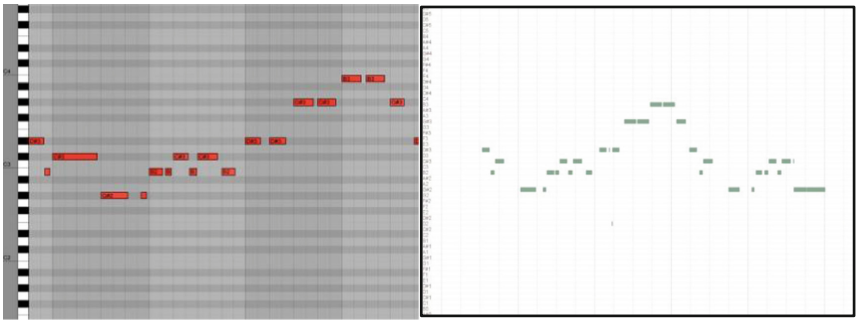
Music information rendering. Piano roll is divided into notes on the y axis and time/beats in axis x. The display is in the form of rectangular shapes which grow depending on the duration of each note in real-time. The rendering also executed through the main thread, and its implementation contains snapshots instances draw on the canvas

3 Evaluation and Perspectives

The application tested in real use cases with different instruments and sources. The tests were both system and subsystem oriented. Some tests achieved with the musical instrument recorded under ideal studio conditions without any external noise at all, and others with a regular microphone on a PC.

The first evaluation has been done on each note separately, with piano. Due to the time-domain approach of the pitch detection algorithm we see that we have no results until the middle of the second octave (Fig. 2 (a) and (b)). The second evaluation (Table 1) is a comparative evaluation that has been done with two different instruments (Guitar, Trumpet) in two commerce software and Note Recognizer.

We note that the result of the application is sufficiently accurate to the original, especially when the track is played at a slow speed. The results' accuracy depends on the kind of the instrument, with the major differentiation factor to be the amount of



(a) (b)

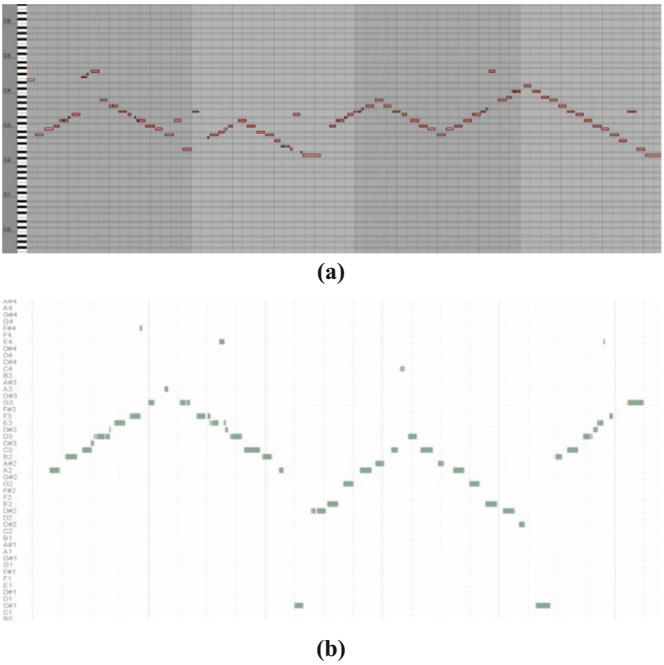
Fig. 2. **(a)** Part of the original notes of the song "Hit the road Jack", **(b)** Capturing part of the song "Hit the road Jack" played with piano at 110 BPM, with redraw corrected functionality

Table 1. Comparative evaluation with two different instruments (Guitar, Trumpet) in two commerce software and Note Recognizer

Instruments	Ableton live 9	WIDI 4.4 Pro	Note Recognizer (real-time)	Note Recognizer (corrected)	Original notes
Guitar	25/25 (+6 errors)	25/25 (+9 errors)	24/25 (+1 error)	24/25 (+1 error)	25
Trumpet	32/32 (+1 error)	31/32 (+15 errors)	32/32 (+3 errors)	32/32 (0 errors)	32

resonance they produce (Fig. 3 (a) and (b)). When the instrument has resonator the application can have numerous adverse results. Also there has been an evaluation of some subsystems. As for Auto-correction functionality we note that it provides correction of the greater percentage of errors that arise at lower frequencies but also discontinuities near or within the notes. The on demand redrawing of the corrected notes is also functioning efficiently. After comparison of our implementation with some professional applications, we noticed that Note Recognizer is quite satisfactory. The application currently has very high computational requirements, especially because of the thread that executes about 50 times per second drawing and onset detection among other functions.

The upcoming version of Note Recognizer covers the performance issue (currently it implements only two web workers), the pianoroll redrawing, the replacement of AnalyserNodes with AudioWorkerNodes and the offset detection function. There are also some goals regarding the creation of some new subsystems. It would be very efficient, for example, to extend the pianoroll to be interactive, so it is possible for the user to edit the rendered notes, play them back and export them as a MIDI file.



(a)

(b)

Fig. 3. **(a)** Music information recording on Ableton live by playing piece on bouzouki, **(b)** Music information recording on Note Recognizer by playing piece on bouzouki

4 Conclusion

In this paper we presented Note Recognizer, a web browser application for processing, retrieval and demonstration of music information from music recordings or music files. Note Recognizer may assist a student to express his/her musical inspiration and at the same time to understand more about the nature of music. The current version is stable and runs on most recent browsers, desktop or mobile. As an experience with web audio development, Note Recognizer is already a usable tool that can be found online at: http://medialab.teicrete.gr/2015_ptixiakes/note_recognizer/.

References

1. Bello, J.P., Daudet, L., Abdallah, S., Duxbury, C., Davies, M., Sandler, M.B.: A tutorial on onset detection in music signals. IEEE Trans. Speech Audio Process. **13**(5), 1035–1047 (2005)
2. Rosão, C., Ribeiro, R., Matos, D.M.: Comparing onset detection methods based on spectral. In: Proceedings of the Workshop on Open Source and Design of Communication, pp. 71–78. ACM (2012)
3. Man, T.K.: Tempo Extraction using the Di. Hong Kong University of Science and Technology, Hong Kong (2006)

4. Cai, W.: Analysis of Acoustic Feature Extraction. University of Rochester, New York (2013)
5. Cnx. (n.d.) (2017). http://cnx.org/contents/8b900091-908f-42ad-b93d-806415434b46@2/Pitch-Detection-Algorithms
6. Bello, J.P., Duxbury, C., Davies, M., Sandler, M.: On the use of phase and energy for musical onset detection in the complex domain. IEEE Signal Process. Lett. **11**(6), 553–556 (2004)
7. Hess, A.: Beat Detection for Automated Music Transcription: An exploration of Onset Detection Algorithms. MSc thesis, Thomas J. Watson School of Engineering and Applied Science State University of New York at Binghamton (2011)
8. The future of music technology (2017). http://www.yalescientific.org/2012/03/the-future-of-music-technology/. Accessed 2 June 2017
9. The future of music technology (2017). http://spectrum.ieee.org/tech-talk/consumer-electronics/audiovideo/the-future-of-music-technology. Accessed 2 June 2017
10. Bayesian Music Transcription: A. Taylan Cemgil Radboud, PHD Thesis, University of Nijmegen, Netherlands (November, 2004) (2004)
11. Klapuri, A., Davy, M. (eds.): Signal Processing Methods for Music Transcription. Springer, New York (2006). https://doi.org/10.1007/0-387-32845-9
12. Bello, J.P., Sandler, M.: Phase-based note onset detection for music signals. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-03), Hong Kong, 6–10 April 2003
13. Beat detection using web audio (2017). http://joesul.li/van/beat-detection-using-web-audio/. Accessed 2 June 2017
14. Goto, M.: A predominant-f0 estimation method for polyphonic musical audio signals. In: 18th International Congress on Acoustics, pp. 1085–1088 (2004)
15. Implementation of Pitch Detection (2017). https://github.com/cwilso/pitchdetect. Accessed 2 June 2017
16. Wikipedia. (n.d.) (2017). https://en.wikipedia.org/wiki/Autocorrelation. Accessed 2 June 2017
17. Implementation of the Moving Median filter (2017). https://github.com/mikolalysenko/moving-median. Accessed 2 June 2017