Scorpiano – A System for Automatic Music Transcription for Monophonic Piano Music

Bojan Sofronievski and Branislav Gerazov
Faculty of Electrical Engineering and Information Technologies
Ss Cyril and Methodius University in Skopje, Macedonia
bojan.sof@hotmail.com, gerazov@feit.ukim.edu.mk

***Abstract*—Music transcription is the process of transcribing music audio into music notation. It is a field in which the machines still cannot beat human performance. The main motivation for automatic music transcription is to make it possible for anyone playing a musical instrument, to be able to generate the music notes for a piece of music quickly and accurately. It does not matter if the person is a beginner and simply struggles to find the music score by searching, or an expert who heard a live jazz improvisation and would like to reproduce it without losing time doing manual transcription. We propose Scorpiano – a system that can automatically generate a music score for simple monophonic piano melody tracks using digital signal processing. The system integrates multiple digital audio processing methods: notes onset detection, tempo estimation, beat detection, pitch detection and finally generation of the music score. The system has proven to give good results for simple piano melodies, comparable to commercially available neural network based systems.**

***Index Terms*—automatic music transcription, musical note recognition, pitch detection, onset detection, audio processing**

I. Introduction

Automatic music transcription (AMT) is a process that automatically identifies the performed notes in a given melody track, with the goal of generating a music score. Besides automatic generation of music scores, AMT applications include interactive music systems and automated music tutors that teach playing an instrument [1], [2].

One of the main determinants for the difficulty of the task of AMT is whether the music to be transcribed is monophonic or polyphonic. Monophonic music simply means that the performer plays only one note at a time. There must not be sounds which overlap and this sounds are characterized by only one pitch. Contrary to monophonic music, polyphonic music consists of two or more simultaneous lines of independent melodies. This means that multiple notes are played at the same time and this sound is characterized with multiple pitches. AMT for polyphonic music is a non-trivial task and is a very active field of research [2]–[4].

There are few different approaches to AMT. One approach is based purely on digital signal processing and it is mainly used for monophonic music. These systems basically integrate methods for pitch and onset/beat detection. There are mainly two approaches for pitch detection: *i*) the time-domain approach, which is primarily based on the autocorrelation function and its modifications [5], [6], *ii*) the frequency-domain approach, which is primarily based on the STFT (Short Time Fourier Transform) [7]. The onset detection algorithms are based on finding the peaks of a novelty function, i.e., a function whose peaks should coincide, within a tolerance margin, with note onset times [8].

The second approach is based on machine learning algorithms, mainly neural networks and these systems are the main choice for transcribing polyphonic music [9]. There are a few commercially available applications which follow this approach, such as AnthemScore[1] and Melody Scanner[2], and they advertise over 80% accuracy. However, neural networks in general require more processing power and a big set of data for training.

We propose the system for AMT of monophonic music, based on the digital signal processing approach, named Scorpiano. The piano is chosen to be the source instrument for transcription, because it produces sound by hitting the strings with hammers, giving the piano better frequency stability of the pitch, compared to other string instruments which produce sounds by plucking the strings. Our systems generates scores with high accuracy and fast transcription speeds, and is comparable to commercially available neural network based systems. The system also only has a small number of parameters that can be easily adjusted for different piano sources. Scorpiano is made available as free software.[3]

II. System Architecture

Fig. 1 shows Scorpiano's system architecture. The input to the system is an audio file of a monophonic piano recording. The system consists of five modules for: onset detection, tempo estimation, beat detection, pitch detection and music score generation. The system outputs an image file of the generated music score.

A. Onset Detection

Onset detection is the task of determining the starting times of musical notes in a music recording. Onsets correspond to a

[1]https://www.lunaverus.com
[2]https://melodyscanner.com
[3]https://gitlab.com/BojanSof/scorpiano

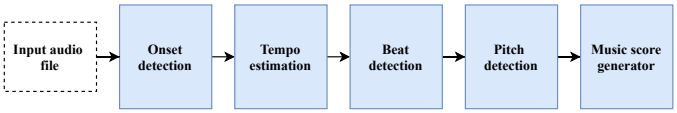

Fig. 1. System architecture.

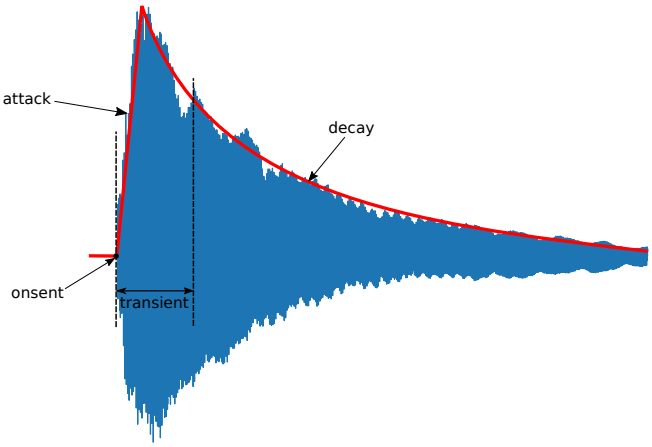

Fig. 2. The onset, attack, transient, and decay of an isolated note.

sudden increase of energy at the beginning of musical notes. There are a number of closely related concepts defined for each note realisation [10], as illustrated in Fig. 2:

- the *onset* of the note is the time moment when the transient starts,
- the *attack* of the note is the time interval during which the amplitude envelope increases, and
- the *transient*, in the case of the piano or other acoustic instruments, corresponds to the period during which the excitation is applied and then damped, leaving only the slow decay at the resonance frequency of the body of the instrument, and
- the *decay* is defined as the time interval in which the amplitude decreases gradually until the sound vanishes.

To detect a sudden increase of the energy, we compute the energy novelty function. The energy novelty function is a function that describes local changes in signal energy. To compute the novelty function, first we compute the local energy function of the signal x, using a bell-shaped window function w with length $2M+1$, e.g. a Hann window, given with:

$$\begin{aligned} E[n] &= \sum_{m=-M}^{m=M} |x[n+m]w[m]|^2 \\ &= \sum_{m\in Z} |x[m]w[n-m]|^2 \\ &= x^2[n] * w^2[n] \end{aligned}$$

To compute the changes of the energy, we take the first derivative of the local energy, which in the discrete case can be approximated by taking the difference between subsequent energy values. Before taking the derivative, we apply logarithmic compression to the local energy function, taking into account the fact that human perception of sound intensity is logarithmic. Because we are interested only in energy increases, we apply a half-wave rectification of the derivative. The half-wave rectification function is given by:

$$r_{\text{half}}(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases}$$

Thus, we obtain the local energy novelty function:

$$E_{\text{novelty}}[n] = r_{\text{half}}\left(E_\gamma[n+1] - E_\gamma[n]\right)$$

where $E_\gamma[n]$ is the logarithmic compression of $E[n]$ for a positive constant γ:

$$E_\gamma[n] = \log\left(1 + \gamma \cdot E[n]\right)$$

Fig. 3 shows the local energy, the energy novelty function and the marked maxima of the novelty function for a part of "Twinkle twinkle little star" melody, using a Hann window with a window length of 46 ms. Note that the energy novelty function is normalised by dividing it with its maximum value.

To detect onsets, we mark the local maxima of the energy novelty function. A simple method for finding local maxima is employed, keeping only maxima above a set amplitude threshold, and discarding maxima that are two close together. We set the amplitude threshold to be 0.1, and the time threshold to be 0.1 s.

B. Tempo Estimation

To find the correct timing of the musical notes and their beat duration, we must estimate the tempo of the music. For this purpose, we use the `beat.tempo` function from *librosa*[4] library [11] to obtain the beats per minute value for the melody.

C. Beat Detection

After detecting the onsets of the musical notes and estimating the tempo, the beat duration of each note should be calculated. For every note, except the last, assuming there are no breaks, note duration can be computed by taking the difference between two subsequent onset moments. Then, the beat duration of the note is computed by multiplying the note duration with the tempo. To find the duration of the last note we can simply track backwards the normalised energy of the signal which represents the last note, and estimate the end of the note using a threshold.

D. Pitch Detection

According to the Fourier representation, each musical sound is a weighted sum of an infinite number of sinusoidal components. The frequency values of this components are integer multiples of the first one, called the fundamental frequency, denoted as F_0, which is perceived as pitch.

We can apply pitch detection using the notes' starting and ending times determined by their onset. We use the YIN

[4]https://librosa.org

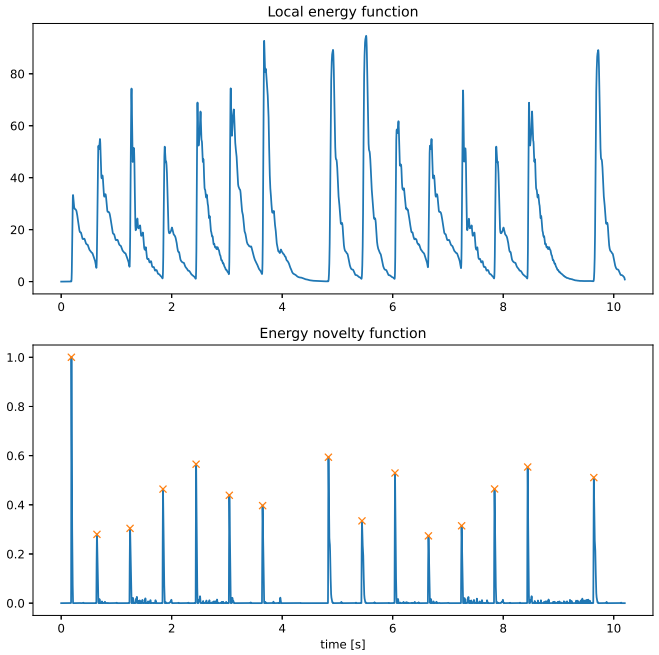

Fig. 3. Local energy function and energy novelty function.

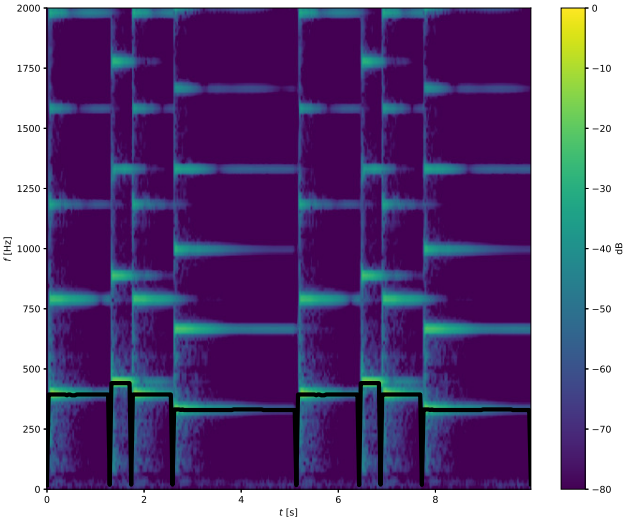

Fig. 4. Pitch contour obtained with the YIN algorithm for part of "Silent Night"

algorithm for pitch detection [6]. It is an improved autocorrelation method for pitch detection, with a few modifications to minimize the error. The YIN algorithm can be sublimed in 6 steps:

1) calculate the autocorrelation function:

$$r_t[\tau] = \sum_{j=t+1}^{t+M-\tau} x[j]x[j+\tau]$$

2) calculate the difference function, over a window with size M samples (corresponding to 68 ms window length in our case):

$$d_t[\tau] = \sum_{j=1}^{M} (x[j] - x[j+\tau])^2$$

which can be expressed using the autocorrelation function as:

$$d_t[\tau] = r_t[0] + r_{t+\tau}[0] - 2r_t[\tau]$$

3) calculate the cumulative mean normalized difference function:

$$d'_t[\tau] = \begin{cases} 1, & \text{if } \tau = 0 \\ \dfrac{d_t[\tau]}{(1/\tau)\sum_{j=1}^{\tau} d_t[j]}, & \text{otherwise} \end{cases}$$

4) threshold – set the absolute threshold and choose the smallest value of τ that gives minimum of the cumulative mean normalized difference function, smaller than the threshold,
5) calculate parabolic interpolation – each local minimum of the cumulative mean normalized difference function and its neighbors is fit by a parabola and the abscissa of the interpolated minimum is used as the period estimate.
6) calculate the best local estimate – repeat the period estimation in a shrinking time interval.

To estimate the fundamental frequency, the median of the estimated pitch periods for each window is computed, due to the fact that the median is more immune to single or few erroneous extreme values in the list of numbers, compared to the average of the numbers.

Figure 4 shows the pitch contour obtained with the YIN algorithm for part of the "Silent Night" melody. The peaks at beginning and ending of each note show why the fundamental frequency is computed using the median of the estimated pitch periods for each window.

E. Music Score Generator

Using the obtained beat durations and pitch of the musical notes, we can generate the music score. The library used for this step is *music21*[5], with the *LilyPond* engine[6]. The time signature of the score must be given by the user. The output is an image file.

III. Experiments

To evaluate the performance of the system, ten popular monophonic piano melodies are chosen.

In the first experiment, the system is tested with the melodies generated using the *MuseScore*[7] application, which allows the generation of audio files from the music score for the melody.

In the second experiment, the system is tested with recordings of melodies played on a Miditech Midistart 3 MIDI keyboard, capable of modelling the dynamics of a piano.

[5]https://web.mit.edu/music21
[6]http://lilypond.org
[7]https://musescore.org

For simulating the piano sound, the *Addictive Keys*[8] software was used and the audio output of the program was recorded using *Audacity*[9]. We will refer to this configuration as "digital piano".

In the third experiment, the effect of the change of tempo on the system accuracy is evaluated with two melodies played on the digital piano, with three different tempos: slow, normal and fast, i.e. 80 bpm, 100 bpm and 120 bpm.

The performance of our system is compared with the commercial *AnthemScore* program. AnthemScore is an AMT software based on neural networks, advertised as being trained on millions of data samples.

To evaluate the performance of our system we define the following three errors:

- **note error rate** – the relative error for the number of original, n_{original}, and detected notes, n_{detected}:

$$\varepsilon_{\text{note}} = \frac{|n_{\text{detected}} - n_{\text{original}}|}{n_{\text{original}}} \times 100\%$$

- **pitch error rate** – the relative error for the number of incorrectly detected pitches, $p_{\text{incorrect}}$, excluding the extra detected notes:

$$\varepsilon_{\text{pitch}} = \frac{p_{\text{incorrect}}}{n_{\text{original}}} \times 100\%$$

- **beat error rate** – the relative error for the number of incorrectly detected beats, $b_{\text{incorrect}}$, excluding the extra detected notes:

$$\varepsilon_{\text{beat}} = \frac{b_{\text{incorrect}}}{n_{\text{original}}} \times 100\%$$

IV. Results

A. MuseScore generated melodies

The number of notes of the original and automatically generated scores, and the number of incorrect pitches and beats excluding the extra detected notes, for each melody, are given in Table I.

The chosen parameter values give nearly perfect results for the chosen melodies. The incorrectly detected beats in the melodies mainly appear for the end note, as shown in Fig. 5 for "London Bridge". The errors in the generated score are marked with red.

From a musical standpoint, the last note marked with red in the generated score, which is tied with the previous note and has the same pitch, means that the duration of the previous note is increased, so the total beat duration of the tie is $2\frac{3}{4}$. In fact, the system detects the last note as one note with duration $2\frac{3}{4}$, but the score generator renders it as a tie.

B. Digital piano recordings

Table II shows the error rates for generated scores using Scorpiano and AnthemScore for the selected melodies. The worst results using Scorpiano are obtained for the "Silent Night" melody. The main reason for this is because the tempo of the melody is estimated to be 143 bpm, which is twice the actual of 70 bpm. Because of this, every note in the generated score has nearly twice the actual beat duration.

The errors in the scores generated with AnthemScore mostly comprise extra notes that are detected as played simultaneously with the correct note, forming a chord. The worst results are for "Canon in D" melody for which AnthemScore estimated the tempo incorrectly to be 113 bpm, versus the actual of 76 bpm. Despite of this, AnthemScore gives really good results for the digital piano recordings, show casing the advantage of using neural networks for AMT. Fig. 6 shows a comparison of the errors in transcription for "Ode to Joy" generated with Scorpiano and AnthemScore. The notes colored red in the generated score with Scorpiano have incorrectly detected pitch. Their fundamental frequency was estimated to be two times smaller than the actual one. This type of error

TABLE I
Comparisons of the original and generated music scores for melodies generated using *MuseScore*.

Melody name	Number of notes (original)	Number of notes (detected)	Incorrect pitches	Incorrect beats
The Alphabet song	43	43	0	0
Auld Lang Syne	58	58	0	1
Cannon in D	46	46	0	0
Happy Birthday	25	25	0	1
Jingle Bells	49	49	0	1
London Bridge	24	25	0	0
Mary had a little lamb	25	25	0	1
Ode to Joy	62	62	0	0
Silent Night	47	47	0	0
Twinkle Twinkle Little Star	42	42	0	0

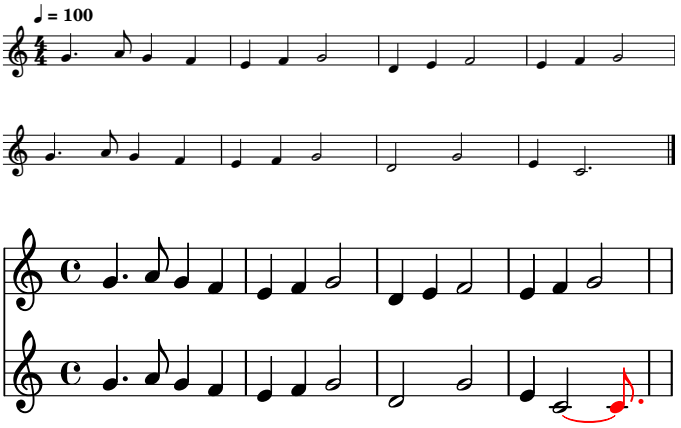

Fig. 5. "London Bridge" original score (top) and generated score (bottom).

[8]https://www.xlnaudio.com/products/addictive_keys
[9]https://www.audacityteam.org

TABLE II
ERROR RATES FOR THE AUTOMATICALLY GENERATED SCORES USING *Scorpiano* AND *AnthemScore* FOR MELODIES PLAYED WITH THE DIGITAL PIANO.

Melody name	Algorithm	Note error rate	Pitch error rate	Beat error rate
Auld Lang Syne	Scorpiano	3.45	1.72	0.00
	AnthemScore	**1.72**	**0.00**	0.00
Canon in D	Scorpiano	**4.35**	0.00	**2.17**
	AnthemScore	8.70	0.00	56.52
London Bridge	Scorpiano	0.00	0.00	16.67
	AnthemScore	0.00	0.00	16.67
Ode to Joy	Scorpiano	**0.00**	3.23	**0.00**
	AnthemScore	17.74	**0.00**	1.61
Silent Night	Scorpiano	70.21	0.00	100
	AnthemScore	**8.51**	0.00	**0.00**

TABLE III
ERROR RATES FOR GENERATED SCORES USING *Scorpiano* FOR MELODIES PLAYED ON A DIGITAL PIANO WITH THREE DIFFERENT TEMPOS.

Tempo	Note error rate	Pitch error rate	Beat error rate
Slow	0.00	0.00	1.02
Normal	0.00	0.00	1.02
Fast	2.38	0.00	0.00

in pitch detection, when the estimated fundamental frequency and the actual one form a ratio equal to a power of 2 is known as *octave error* [12].

C. Tempo effect

Table III shows the average error rates for generated scores with Scorpiano, for the melodies "Jingle Bells" and "Twinkle Twinkle Little Star" played on a digital piano with three different tempos: slow, normal and fast. From the results, it can be concluded that overall, the tempo has a small effect on the system performance, although there are cases, like with the "Silent Night" melody in the digital piano experiment, where the incorrect estimation of the tempo makes beat duration of the notes incorrect. This problem also happened with AnthemScore in the digital piano experiment for "Canon in D" melody.

D. Summary

Table IV summarises the average error rates for Scorpiano and AnthemScore for each experiment. Both systems score equally good for the MuseScore and Tempo experiment. AnthemScore shows slightly better results for the Digital piano experiment.

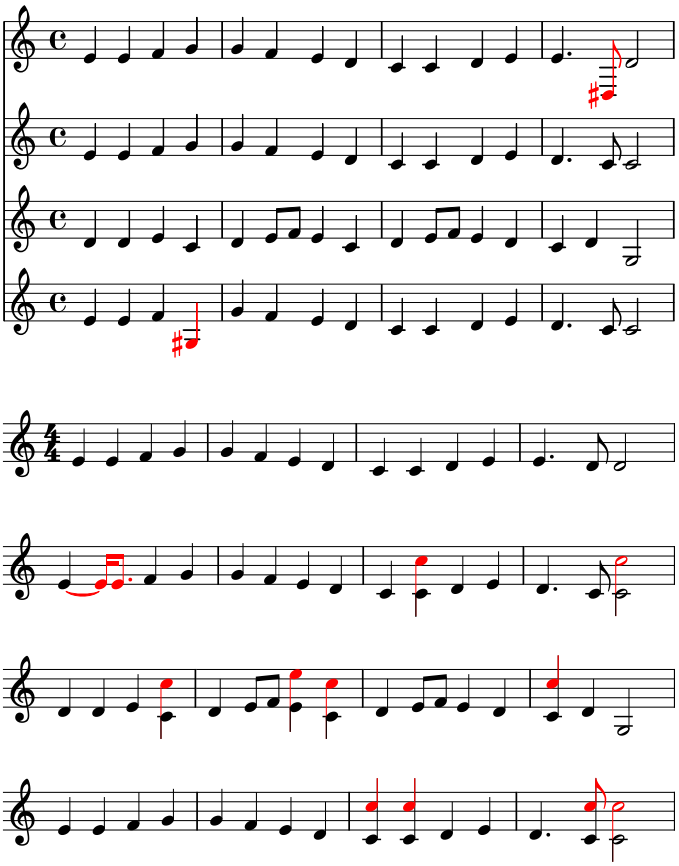

Fig. 6. Scorpiano (top) and AnthemScore (bottom) automatically generated scores for "Ode to Joy" played with a digital piano.

V. CONCLUSION

The problem of AMT for monophonic music can be effectively addressed using digital processing techniques. We propose a AMT system for monophonic piano music, that uses pure digital signal processing and has the advantage of being computationally inexpensive, fast, and does not need big training sets, whilst obtaining good results with low error rates, comparable to commercial neural network based systems.

In its current form, the system lacks detection of breaks and

TABLE IV
SUMMARY RESULTS.

Experiment	Algorithm	Note error rate	Pitch error rate	Beat error rate
MuseScore	**Scorpiano**	**0.42**	**0.00**	**1.18**
	AnthemScore	0.99	0.00	1.49
Digital piano	Scorpiano	8.20	0.89	12.89
	AnthemScore	**5.94**	**0.00**	**8.35**
Tempo	**Scorpiano**	**0.79**	**0.00**	**1.02**
	AnthemScore	1.70	0.00	1.19

time signatures. Sometimes mistakes can make the generated score look wrong, although most of the time the errors were obvious and could easily be corrected by human intervention.

The described system can be extended in the future. Post-processing the outputs of the modules can improve the performance of the system. For example, the estimated fundamental frequency for each note can be compared with the neighbouring notes to avoid octave errors. The system can also be modified to work with different musical instruments.

REFERENCES

[1] E. Benetos, S. Dixon, D. Giannoulis, H. Kirchhoff, and A. Klapuri, "Automatic music transcription: Challenges and future directions," *Journal of Intelligent Information Systems*, vol. 41, 12 2013.

[2] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic music transcription: An overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2019.

[3] G. Costantini, M. Todisco, and G. Saggio, "Automatic music transcription based on non-negative matrix factorization," 2010.

[4] J. Sleep, "Automatic music transcription with convolutional neural networks using intuitive filter shapes," 10 2017.

[5] P. S. Rao, S. Khoushikh, S. Ravishankar, R. A. Ananthkrishnan, and K. Balachandra, "A comparative study of various pitch detection algorithms," in *2020 5th International Conference on Computing, Communication and Security (ICCCS)*, 2020, pp. 1–6.

[6] A. de Cheveigné and H. Kawahara, "Yin, a fundamental frequency estimator for speech and music," *Acoustical Society of America*, vol. 13, Apr. 2002.

[7] B. Gerazov and Z. Ivanovski, "Building a basis for automatic melody extraction from macedonian rural folk music," 06 2010.

[8] C. M. T. Rosão, "Onset detection in music signals," Ph.D. dissertation, University of Lisbon, 2012. [Online]. Available: http://hdl.handle.net/10071/5991

[9] F. Saputra, U. G. Namyu, Vincent, D. Suhartono, and A. P. Gema, "Automatic piano sheet music transcription with machine learning," *Journal of Computer Science*, vol. 17, no. 3, pp. 178–187, Mar. 2021. [Online]. Available: https://thescipub.com/abstract/jcssp.2021.178.187

[10] J. P. Bello, L. Daudet, S. Abdallah, C. Duxbury, M. Davies, and M. B. Sandler, "A tutorial on onset detection in music signals," *IEEE TRANSACTIONS ON SPEECH AND AUDIO PROCESSING*, vol. 13, pp. 1035–1047, Sep. 2005.

[11] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, 2015.

[12] B. Kumaraswamy and P. G. Poonacha, "Octave error reduction in pitch detection algorithms using fourier series approximation method," *IETE Technical Review*, vol. 36, no. 3, pp. 293–302, 2019. [Online]. Available: https://doi.org/10.1080/02564602.2018.1465859

[13] M. Mller, *Fundamentals of Music Processing: Audio, Analysis, Algorithms, Applications*, 1st ed. Springer Publishing Company, Incorporated, 2015.

[14] Q. Gao, "Pitch detection based monophonic piano transcription," *Yankee*, vol. 7, no. 60, p. C4, 2015.