IMPROVING MIDI GUITAR'S ACCURACY WITH NMF AND NEURAL NET

Masaki Otsuka and **Tetsuro Kitahara**
Graduate School of Integrated Basic Sciences, Nihon University
{masaki,kitahara}@kthrlab.jp

ABSTRACT

In this paper, we propose a method for improving the accuracy of MIDI guitars. MIDI guitars are useful tools for various purposes from inputting MIDI data to enjoying a jam session system, but existing MIDI guitars do not have sufficient accuracy in converting the performance to an MIDI form. In this paper, we make an attempt on improving the accuracy of a MIDI guitar by integrating it with an audio transcription method based on non-negative matrix factorization (NMF). First, we investigate an NMF-based algorithm for transcribing guitar performances. Although the NMF is a promising method, an effective post-process (i.e., converting the NMF's output to an MIDI form) is a non-trivial problem. We propose use of a neural network for this conversion. Next, we investigate a method for integrating the outputs of the MIDI guitar and NMF. Because they have different tendencies in wrong outputs, we take an policy of outputting only common parts in the two outputs. Experimental results showed that the F-score of our method was 0.626 whereas those of the MIDI-guitar-only and NMF-and-neural-network-only methods were 0.347 and 0.526, respectively.

1. INTRODUCTION

A MIDI guitar, which outputs the user's performance data into the MIDI format in real time, is useful for guitarists to engage in various music activities such as inputting MIDI data into a computer and enjoying the use of a jam session system. However, the accuracy of MIDI guitars is not as high as a MIDI keyboard because the MIDI guitar detects the strings' vibration by analyzing the temporal changes in the magnetic field around the strings.

There have been many attempts made to transcribe guitar performances [1–3, 5, 8–10, 12]. Arimoto et al. remade the PreFEst method, originally developed by Goto [4], for a guitar based on physical constraints on fingering forms [1]. Yazawa et al. also focused on latent harmonic allocation for a guitar based on physical constraints in fingering form [12]. Barbancho et al. furthermore investigated these physical constraints [2]. Fiss et al. constructed a system that transcribes a guitar performance as a tablature [3]. This system estimates not only the notes that are played, but it also estimates how the notes are played (i.e., string number and fret number) through audio signal processing. O'Grady et al. considered both the use of non-negative matrix factorization (NMF) [7] and a hardware improvement of a MIDI guitar for accurate guitar performance transcription [8]. Harquist also proposed a real-time guitar transcription method using NMF [5]. In addition, there have been attempts to improve guitar performance transcription by integrating audio signal processing with computer vision [9, 10].

In this paper, we focus on improving the accuracy of MIDI guitars by integrating them with audio signal processing technologies (especially NMF). Almost all MIDI guitars have an audio output jack for connecting to a guitar amplifier as well as a MIDI output. By connecting this audio output jack to a PC's audio input jack, the guitar's audio signal can be analyzed. By inputting the guitar's MIDI output to that PC, the audio result and the guitar's MIDI output can be integrated. Thus, introducing audio signal processing to a MIDI guitar does not require any special equipment or hardware improvements. O'Grady et al. focused on a similar technique as we employ here but their method involved hardware improvements [8]; our methodology requires no hardware improvement. Using computer vision [9, 10] is an interesting approach, but it requires installing a camera and it may restrict the player's motions. Using physical constraints in fingering forms [1, 2, 12] is a common and promising approach, but we dare to adopt the approach of exploring how much we can improve the accuracy without using physical constraints. Physical constraints can be applied to our method for further improvements in the future.

The rest of this paper is organized as follows: In Section 2 we propose a method for transcribing guitar performance using NMF. In Section 3 we describe a method for integrating NMF-based transcription outputs and the MIDI guitar's outputs. In Section 4 we report our experimental results. Finally, we conclude the paper in Section 5.

2. AUDIO-TO-MIDI CONVERSION WITH NMF

NMF is a technique for decomposing a matrix V into the product of two matrices W and H, that is, $V \cong WH$, where W is a basis matrix and H is a gain matrix. A typical usage of NMF in automatic music transcription is to apply NMF to a spectrogram. Then, the basis matrix W is an array of N column vectors $\boldsymbol{w}_n$ that represent the spectrum of each note n; the gain matrix is an array of N row vectors $\boldsymbol{h}_n$ that represent a temporal sequence of the gain for the basis vector $\boldsymbol{w}_n$. Because the gain vector $\boldsymbol{h}_n$ represents an approximation of a temporal sequence of the amplitude for the note n, the onset and offset times for a note n can be identified by thresholding $\boldsymbol{h}_n$; steep rises in the time series $\{h_{n,t}\}_t$ represent onsets and steep drops represent offsets.

However, there are two problems with this technique. The first one is that standard NMF is applicable only after the entire spectrogram is obtained. This fact means that standard NMF cannot be used for real-time processing. The second problem is that it is difficult to determine a universally appropriate threshold because the actual gains vary according to playing style, strings, and other factors. Thus, the issues to be resolved here can be summarized as follows:

Issue 1 How to apply NMF to real-time processing

Issue 2 How to determine an appropriate threshold depending on the playing style, strings, etc.

In this paper, we resolve these issues as follows:

Solution 1 We ask the user to play a chromatic scale (from the lowest note to the highest note) for each string in advance and apply NMF to this *preliminary performance*. We assume that the spectrum of each note is similar enough between the preliminary and target performances[1] if the same person plays the same instrument in the same way. Under this assumption, the basis matrix calculated from the preliminary performance is then used to obtain the gain vectors for the target performance.

Solution 2 We introduce one more preliminary performance and adaptively determine the threshold. This preliminary performance has a similar musical feature to the target performance, and we ask the user to play the phrase specified by the system accurately (thus, the system knows the ground truth). Adaptation of the threshold using these data is approximately equivalent to learning a neural network. We therefore learn how high the gain is and how steeply the gain rises at onsets with a neural network and use this neural network for detecting onsets.

In the rest of this section, we first describe a method in which only Solution 1 is introduced (we call this method the *baseline method*). Next, we introduce Solution 2 to this baseline method.

[1] The *target performance* refers to the performance to be converted to the MIDI format.

2.1 Baseline method — Introducing Solution 1 only

Stage 1: Estimating basis matrix from preliminary performance

After the user plays all chromatic notes successively for each string k (called the *1st preliminary performance*), the spectrogram V_k is calculated using the short-term Fourier transform with a 4096-point Hamming window shifted by 10 ms (we suppose 44.1-kHz sampling). Then, the spectrogram V_k is decomposed into the basis matrix W_k and the gain matrix H_k using NMF. To avoid that spectral peaks for different notes are mixed into a single basis vector, we prepare 35 basis vectors for each string even though each string has 23 notes. We then obtain 23 basis vectors by merging pairs of basis vectors that have a high cosine similarity.

Stage 2-1: Estimating gain vectors for target performance

The user plays the target performance (i.e., the performance to be converted to the MIDI format). As the user plays, the power spectrum $\boldsymbol{v}_t$ (where t is time) is obtained via the Fourier transform, and then the gain vector $\boldsymbol{h}_{t,k}$ for each string k is calculated. The gain vector $\boldsymbol{h}_{t,k}$ is defined as $\boldsymbol{h}_{t,k} = W_k^{-1}\boldsymbol{v}_t$, where W_k is the basis matrix for the string k, obtained in Stage 1. Because W_k is not a square matrix in general, its inverse matrix cannot be calculated. We therefore use a pseudo-inverse matrix [6] instead.

Stage 2-2: Generating MIDI messages by thresholding gain vectors

After the gain vector $\boldsymbol{h}_{t,k}$ is calculated, MIDI messages are generated. When the n-th element of $\boldsymbol{h}_{t,k}$ has a higher value than the threshold h_0 but that of $\boldsymbol{h}_{t-1,k}$ does not (that is, $h_{t-1,k,n} \leq h_0 < h_{t,k,n}$), a MIDI Note On message for the note number corresponding to fret n of string k is generated. When $h_{t-1,k,n} > h_0 \geq h_{t,k,n}$, a MIDI Note Off message is generated. When $h_{t-2,k,n} > h_{t-1,k,n}$ and $h_{t-1,k,n} < h_{t,k,n}$, even if both $h_{t-1,k,n}$ and $h_{t,k,n}$ are higher than h_0, we can consider that a note is played again before the previous note is decayed enough. If this is the case, a Note Off message is generated at time $t-1$ and a Note On message at time t.

2.2 Introducing Solution 2

The method discussed above involves thresholding but the appropriate threshold depends on various factors including the individual instrument, the strength of picking, and the characteristics of the player. In practice, dynamic adjustment of the threshold is not straightforward. We therefore add one more preliminary performance (called the *2nd preliminary performance*) and adjust the threshold using this 2nd preliminary performance under the assumption that a correct transcription of the 2nd preliminary performance has been given. Let $h_{t,k,n}$ be the gain in the 2nd preliminary performance at time t, string k, and fret n. Whether t is an onset time at fret n of string k in this peformance

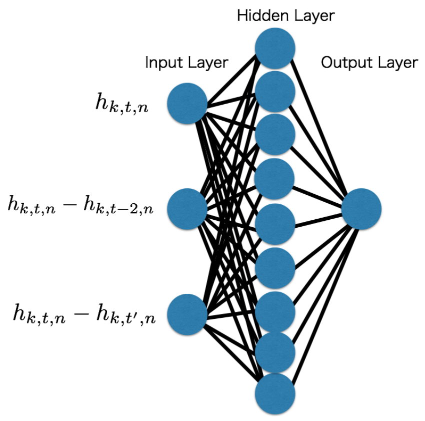

Figure 1. Neural network that we employ

can be identified from the correct transcription, then it is represented as follows:

$$s_{t,k,n} = \begin{cases} 1 & (t \text{ is an onset time at fret } n \text{ of string } k) \\ 0 & (\text{else}) \end{cases}$$

What to be solved here is to find h_0 such that $h_{t,k,n} > h_0$ iff $s_{t,k,n} = 1$. This h_0 can be estimated by minimizing $E(h_0) = \sum_{t,k,n}\{\varsigma(-h_0 + h_{t,k,n}) - s_{t,k,n}\}^2$, where $\varsigma(x)$ is the sigmoid function, that is, $\varsigma(x) = 1/(1 + e^{-x})$. It is equivalent to training a neural network. The temporal differential of $h_{t,k,n}$ is also considered important for onset detection, so we obtain a neural network shown in Figure 1 by adding such features.

Stage 1: Estimating basis matrix from 1st preliminary performance

In the same way as Stage 1 of the baseline method, the 1st preliminary performance is played by the user, and then the basis matrix W_k for each string k is calculated.

Stage 2-1: Estimating gain vectors for 2nd preliminary performance

The user plays the 2nd preliminary performance. As he/she plays, the power spectrum $\boldsymbol{v}_t$ and the gain vector $\boldsymbol{h}_{t,k} = W_k^{-1}\boldsymbol{v}_t$ for each string k are calculated every 10 ms in the same way as in Stage 2-1 of the baseline method.

Stage 2-2: Learning neural network

For each element $h_{t,k,n}$ of $\boldsymbol{h}_{t,k}$, the following steps are performed if $h_{t,k,n}$ is a peak:

1. Features are extracted from $h_{t,k,n}$ and are set to the vector $\boldsymbol{x}_{t,k,n}$. In the current implementation, the following feature vector is used:

   $$\boldsymbol{x}_{t,k,n} = (h_{t,k,n},\ h_{t,k,n} - h_{t-2,k,n},\ h_{t,k,n} - h_{t',k,n}),$$

   where t' is the time of the last valley before t in $\{h_{\tau,k,n}\}_{\tau=0,\cdots,t}$, in other words, the maximum value of τ ($< t$) such that $h_{\tau,k,n} < h_{\tau-1,k,n}$ and $h_{\tau,k,n} < h_{\tau+1,k,n}$.

2. The supervision $s_{t,k,n}$ is defined as described above.

3. The neural networks shown in Figure 1 are trained using backpropagation such that the difference between the value of the output node $y_{t,k,n}$ and the supervison $s_{t,k,n}$ is minimized. We prepare and train different neural networks for different string, but we use the same neural network for different frets of the same string due to a limited number of training data. Each neural network has a single hidden layer consisting of three to ten nodes (we try all cases and present the best result).

In the training, the number of data with supervisions of 1 and 0 are balanced.

Stage 3-1: Estimating gain vectors for target performance

During the target performance, the spectrum and the gain vector are calculated every 10 ms in the same way as in Stage 2-1.

Stage 3-2: Generating MIDI messages based on neural network

The feature vector $\boldsymbol{x}_{t,k,n}$ is calculated in the same way as in Stage 2-2. Then, the value of the output node $y_{t,k,n}$ in the trained neural network is calculated for each time, each string, and each fret. When this value is higher than 0.5, a MIDI Note On message for the corresponding note number is generated.

Theoretically, offsets can also be learned and estimated with a neural network. However, for simplicity, offsets are detected in the same way as in the baseline method.

3. INTEGRATION OF MIDI GUITAR AND NMF

In this section, we describe a method for integrating the outputs of a MIDI guitar and the method discussed in Section 2.2. When we discuss how to integrate two different outputs, we should consider a tradeoff between recall rates and precision rates. We believe that precision is more important in our task because false positives (MIDI messages generated but actually not played) directly result in dissonant sound; false negatives (MIDI messages not generated but actually played) do not. We therefore adopt an approach of outputting the common part of the two outputs.

Stages 1 to 3-2

We perform the same process as in Section 2.2 is performed until Stage 3-2. Although in the method in Section 2.2 the value of the output node $y_{t,k,n}$ is thresholded, it is not thresholded here because $y_{t,k,n}$ is used in Stage 4.

Stage 4: Integration with MIDI guitar outputs

From the output of the MIDI guitar, we obatin the following value:

$$m_{t,k,n} = \begin{cases} 1-\alpha & \text{(the note corresponding to fret } n \text{ of string } k \text{ is being played.)} \\ \alpha & \text{(else)} \end{cases}$$

"(A note is) being played" means the state in which the MIDI guitar had output a MIDI Note On message for this note number but has not yet output a MIDI Note Off message. Note that it represents the MIDI guitar's estimation, so it may not agree with whether that note is actually played. In the equation above, α is a parameter and is set to 0.3 in the current implementation.

Then, $z_{t,k,n} = m_{t,k,n} y_{t,k,n}$ is calculated. The guitar can play only one note at each string at the same time. As a result,

$$\hat{n}_{t,k} = \operatorname*{argmax}_n z_{t,k,n}$$

is calculated and the fret $\hat{n}_{t,k}$ of string k is considered to be played at time t. Then, a MIDI Note On message is generated for the corresponding note number. However, no fret is considered to be played on string k at time t when every element of $\{z_{t,k,n}\}_n$ is lower than a certain threshold (0.3 in the current implementation).

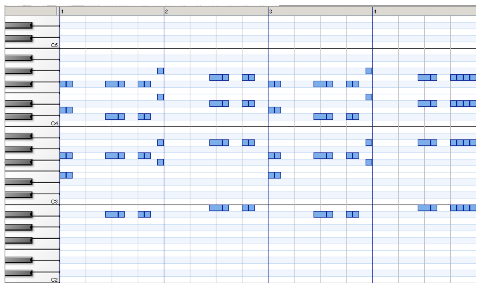
Figure 2. Phrase 1 for the 2nd preliminary performance

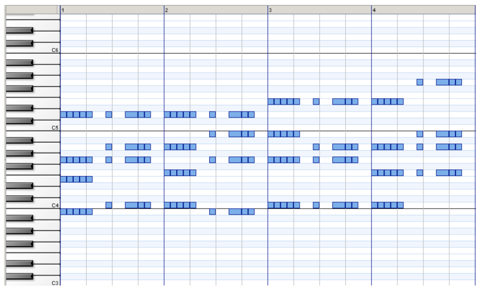
Figure 3. Phrase 2 for the 2nd preliminary performance

4. EXPERIMENTS

4.1 Experiment 1 — Use of neural network

Experimental conditions

To confirm the effect of the use of the neural network described in Section 2.2, we conducted an experiment about converting guitar performances to the MIDI format. We used a Roland GK-3 installed into a Stratocaster as a MIDI guitar with a guitar synthesizer (Roland GR-55). The first author of this paper played 79 four-measure funk rhythmic phrases taken from a guitar phrase book [11]. Of these 79 phrases, those shown in Figures 2 and 3 were used for the 2nd preliminary performance. The remaining phrases were used for test data. We attempted the following three cases:

Case 1 Using only Figure 2,

Case 2 Using only Figure 3, and

Case 3 Using both Figures 2 and 3

for the 2nd preliminary performance. We used neural networks that had three to ten nodes in the hidden layer and will present only the best result.

Experimental results

The results are listed in Table 1. The number of hidden nodes was three. For brevity, we list the accuracy for each chapter instead of each phrase in [11]. In [11], phrases are divided into 16 chapters according to their playing styles, and each chapter includes several phrases. Whereas the F-score for the baseline method was 0.516 on average, the F-score for the proposed method was 0.526 in Case 3. This difference is not very large but one must consider that the proposed method acquired the best threshold because the listed result for the baseline method was the best one in given various thresholds.

Figure 4 shows an example of the experimental results. While the baseline method generated many false positives, especially in the first measure, most of these false positives were eliminated by the proposed method. Thus, the precision rate was improved from 0.659 to 0.692. However, some positives were eliminated so the recall rate decreased slightly (from 0.562 to 0.556).

Figure 5 shows another example. Whereas the baseline method generated many false negatives from the beginning to the end, such false negatives were eliminated by the proposed method. The precision rate was improved from 0.465 to 0.661.

4.2 Experiment 2 — Integration

Experimental conditions

To confirm the effect of the integration described in Section 3, we conducted on audio-to-MIDI conversion of guitar performances using the MIDI guitar only (MGT), the NMF and neural network only (NMF+NN; Section 2.2), and their integration (INT; Section 3). We used the same data as Experiment 1. For the 2nd preliminary performance, we used both Figures 2 and 3 (Case 3). Also in this experiment, we used neural networks that had three to ten nodes in the hidden layer, and will present only the best result for each condition.

Experimental results

The results are listed in Table 2. The numbers of hidden nodes were three for NMF+NN and nine for INT. Whereas the precision rates for MGT and NMF+NN were 0.258 and 0.513, respectively, the precision rate improved to 0.660

Table 1. Result of Experiment 1 (R: recall rates, P: precision rates, F: F-score)

	Baeline method			Proposed method								
	(Simple thresholding)			Case 1			Case 2			Case 3		
Chapters	R	P	F	R	P	F	R	P	F	R	P	F
1	0.640	0.509	0.552	0.636	0.422	0.503	0.642	0.376	0.473	0.611	0.458	0.521
2	0.647	0.489	0.555	0.624	0.483	0.538	0.661	0.490	0.563	0.635	0.574	0.595
3	0.579	0.496	0.531	0.468	0.483	0.472	0.603	0.502	0.541	0.525	0.560	0.539
4	0.536	0.510	0.519	0.554	0.509	0.529	0.576	0.479	0.521	0.562	0.524	0.541
5	0.731	0.557	0.585	0.759	0.399	0.503	0.809	0.431	0.550	0.838	0.437	0.561
6	0.538	0.410	0.465	0.531	0.459	0.491	0.581	0.427	0.491	0.539	0.460	0.496
7	0.580	0.601	0.564	0.562	0.624	0.569	0.640	0.569	0.590	0.636	0.668	0.635
8	0.528	0.577	0.540	0.499	0.481	0.488	0.544	0.520	0.528	0.518	0.536	0.525
9	0.401	0.489	0.431	0.415	0.418	0.413	0.425	0.423	0.422	0.416	0.451	0.430
10	0.464	0.403	0.429	0.476	0.346	0.397	0.472	0.326	0.376	0.442	0.350	0.382
11	0.432	0.621	0.505	0.445	0.621	0.510	0.466	0.589	0.516	0.470	0.643	0.535
12	0.313	0.600	0.411	0.343	0.502	0.407	0.373	0.429	0.399	0.399	0.567	0.468
13	0.621	0.527	0.541	0.568	0.505	0.504	0.652	0.522	0.559	0.611	0.541	0.543
14	0.412	0.442	0.422	0.395	0.425	0.407	0.456	0.425	0.436	0.441	0.473	0.451
15	0.576	0.407	0.474	0.460	0.362	0.384	0.558	0.521	0.463	0.520	0.554	0.418
Final	0.475	0.413	0.422	0.480	0.400	0.424	0.485	0.377	0.415	0.484	0.416	0.437
Average	0.530	0.503	0.516	0.513	0.465	0.488	0.559	0.463	0.506	0.540	0.513	0.526

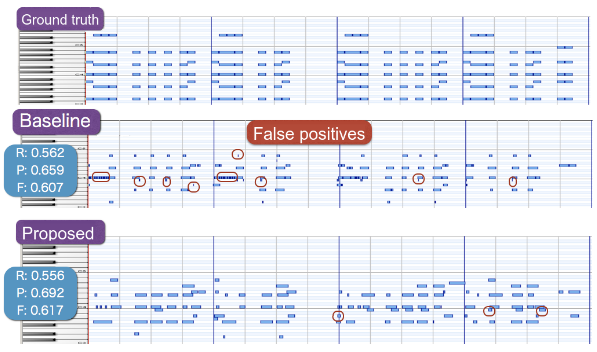

Figure 4. Example of Experiment 1 (Track 47-2)

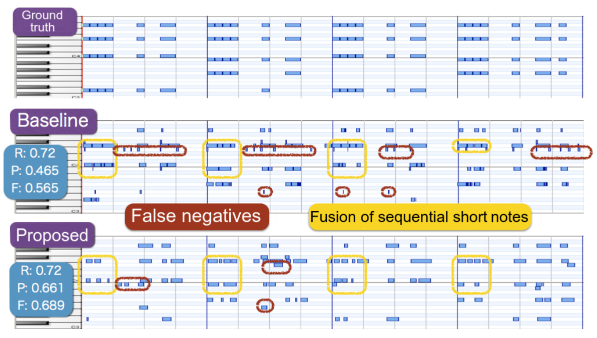

Figure 5. Example of Experiment 1 (Track 09-1)

via the integration. This fact arises because many false positives were eliminated in INT. The recall rate for INT was 0.595; that for MGT was 0.528. The recall rate improved because sequential short notes were fused in MGT, as will be illustrated below, but such errors rarely appeared in INT. Accordingly, the F-score for INT was 0.626; the F-scores for MGT and NMF+NN were 0.347 and 0.526, respectively.

Focusing on the results for each chapter, we can see that the recall rates for 11 chapters (Chapters 1, 2, 3, 4, 6, 7, 8, 10, 12, 13, and final) was improved compared with MGT. However, for other chapters (Chapters 5, 9, 11, 14, and 15) the recall rates decreased. Chapter 5 in [11] features monophonic phrases but the 2nd preliminary performance did not include monophonic phrases. This mismatch is why the recall rate decreased in Chapter 5. The phrases in Chapters 9, 14, and 15 also included monophonic notes.

On the other hand, the precision rate improved for every chapter compared with MGT. In particular, the precision rate improved by more than 0.5 for Chapters 5, 11, and 13.

Figure 6 shows an example of the results. While MGT generated false positives in the whole phrase, such false positives were eliminated in INT as described above. Thus, the precision rate significantly improved from 0.48 (MGT) to 0.83 (INT). At the same time, however, some true positives were also eliminated. On the other hand, MGT caused errors in the fusion of sequential short notes; INT reduced such errors. Eventually, the recall rate increased from 0.60 (MGT) to 0.67 (INT).

Figure 7 shows another example. Similar to the data shown in Figure 6, MGT resulted in errors due to the fusion of sequential short notes at the first beat of every measure. In INT, such errors were corrected. Thus, the recall rate was improved from 0.61 (MGT) to 0.80 (INT). In addition, MGT generated false positives from the second half of the first measure to the last measure; these false positives were eliminated in INT. Thus, the precision rate improved from 0.25 (MGT) to 0.78 (INT). Accordingly, the F-score improved from 0.36 (MGT) to 0.79 (INT).

Table 2. Result of Experiment 2 (R: recall rates, P: precision rates, F: F-score)

	MIDI guitar (MGT)			NMF+NN			Integration (INT)		
Chapters	R	P	F	R	P	F	R	P	F
1	0.668	0.445	0.513	0.611	0.458	0.521	0.758	0.589	0.660
2	0.630	0.230	0.338	0.635	0.574	0.595	0.717	0.648	0.679
3	0.380	0.310	0.327	0.525	0.560	0.539	0.763	0.613	0.679
4	0.505	0.233	0.313	0.562	0.524	0.541	0.508	0.670	0.575
5	0.805	0.110	0.195	0.838	0.437	0.561	0.731	0.643	0.675
6	0.583	0.187	0.287	0.539	0.460	0.496	0.649	0.660	0.641
7	0.493	0.205	0.278	0.636	0.668	0.635	0.597	0.660	0.606
8	0.503	0.215	0.295	0.518	0.536	0.525	0.593	0.644	0.602
9	0.533	0.345	0.408	0.416	0.451	0.430	0.398	0.603	0.469
10	0.463	0.190	0.267	0.442	0.350	0.382	0.732	0.545	0.623
11	0.425	0.345	0.375	0.470	0.643	0.535	0.378	0.825	0.493
12	0.310	0.260	0.285	0.399	0.567	0.468	0.488	0.708	0.574
13	0.438	0.183	0.250	0.611	0.541	0.543	0.640	0.674	0.644
14	0.555	0.280	0.360	0.441	0.473	0.451	0.475	0.776	0.550
15	0.652	0.354	0.434	0.520	0.554	0.418	0.532	0.695	0.517
Final	0.505	0.238	0.313	0.484	0.416	0.437	0.555	0.607	0.542
Average	0.528	0.258	0.347	0.540	0.513	0.526	0.595	0.660	0.626

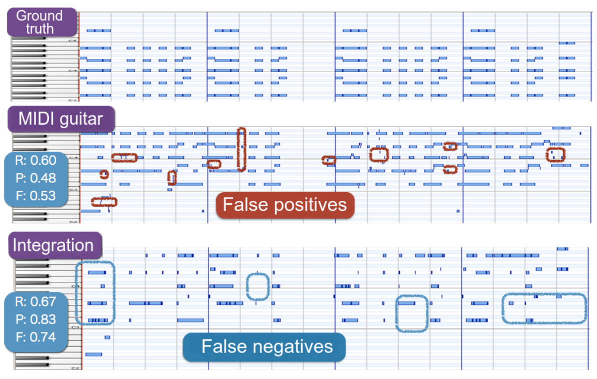

Figure 6. Example of Experiment 2 (Track 47-2)

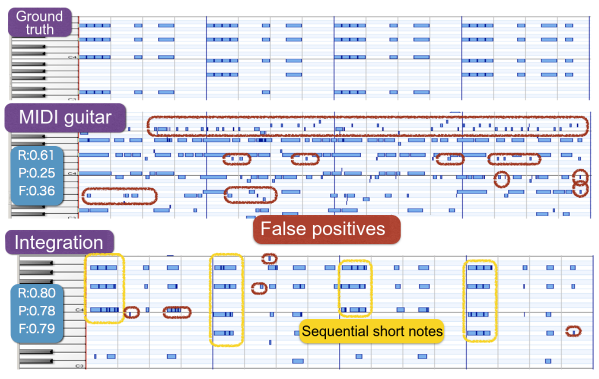

Figure 7. Example of Experiment 2 (Track 09-2)

5. CONCLUSION

A MIDI guitar is a promising tool for guitarists and therefore is being sold by electronic musical instrument manufacturers. However, the audio-to-MIDI conversion accuracy of MIDI guitars is still insufficient. In particular, the accuracy is very low for phrases including many brushing notes like those used in our experiments. To improve this accuracy, we attempted to integrate the output of the MIDI guitar and the signal processing result of the guitar's audio output. Our experimental results showed a significant improvement in accuracy: the F-score was 0.626 compared with 0.347 for the MIDI guitar only.

Although this improvement is significant, we need to improve the accuracy even more to ensure practical use of MIDI guitars. An idea for further improvement may be increasing quantity of training data for the neural network (i.e., the 2nd preliminary performance). However, increasing these data will result in an increase in the user's labor and the time required for learning the neural network. We will therefore investigate a reasonable tradeoff between these time investments and the outcome. In addition, we will assess the latency in outputting MIDI messages because this latency is an important factor in the use of MIDI guitars as musical instruments.

Acknowledgment: This work was supported by JSPS KAKENHI Grant Number 26240025. Also, we thank the members of this JSPS KAKENHI project including Prof. Shigeki Sagayama and Prof. Gen Hori for their fruitful suggestions.

6. REFERENCES

[1] K. Arimoto, T. Fujishima, and M. Goto. A multiple F0 estimation method using specific harmonic structure models for guitar performances (in Japanese). In *Proc. 2006 Autumn Meeting of Acoustic Society of Japan*, pages 585–586. 2006.

[2] A. M. Barbancho and A. Klapuri. Automatic transcription of guitar chords and fingering from audio. In *IEEE Transactions on Audio, Speech, and Language Processing*, volume 20, pages 915–921. 2012.

[3] X. Fiss and A. Kwasinski. Automatic real-time electric gui-

tar audio transcription. In *Proc. IEEE-ICASSP 20111*, pages 373–376. 2011.

[4] M. Goto. A real-time music-scene-description system: Predominant-F0 estimation for detecting melody and bass lines in real-world audio signals. *Speech Comm.*, 43(4):311–329, 2004.

[5] J. Hartquist. Real-time musical analysis of polyphonic guitar audio. Master's thesis, The Faculty of California Polytechnic State University, 2012.

[6] A. B. Israel and T. N. E. Greville. *Generalized Inverses: Theory and Applications*. Springer, 2003.

[7] D. D. Lee and H. S. Seung. Learning the parts of objects with nonnegative matrix factorization. *Nature*, 401:788–791, 1999.

[8] P. D. O'Grady and S. T. Rickard. Automatic hexaphonic guitar transcription using non-negative constraints. In *Proc. IEEE-ISSC 2009*. 2009.

[9] M. Paleari, B. Huet, A. Schutz, and D. Slock. A multimodal approach to music transcription. In *Proc. IEEE-ICIP 2008*, pages 93–96, 2008.

[10] T. Yamagami and K. Itou. A bimodal music dictation method for composition support by using guitar performance video (in Japanese). In *Proc. of IPSJ National Convention 2014*, volume 2, pages 365–366, 2014.

[11] K. Yamaguchi. *16 beat ga minitsuku! Funk de oboeru otona no cutting (in Japanese)*. Rittor Music, 2013.

[12] K. Yazawa, K. Itoyama, and H. G. Okuno. Automatic transcription of guitar tablature from audio signals in accordance with player's proficiency. In *Proc. IEEE-ICASSP 2014*, pages 3146–3150. 2014.