Electronic Preprint for Journal of Information Processing Vol.24 No.3

Regular Paper

LiVo: Sing a Song with a Vowel Keyboard

Kazuhiko Yamamoto[1,a)] Takeo Igarashi[1,b)]

Received: June 15, 2015, Accepted: January 12, 2016

Abstract: We propose a novel user interface that enables control of a singing voice synthesizer at a live improvisational performance. The user first registers the lyrics of a song with the system before performance, and the system builds a probabilistic model that models the possible jumps within the lyrics. During performance, the user simultaneously inputs the lyrics of a song with the left hand using a vowel keyboard and the melodies with the right hand using a standard musical keyboard. Our system searches for a portion of the registered lyrics whose vowel sequence matches the current user input using the probabilistic model, and sends the matched lyrics to the singing voice synthesizer. The vowel input keys are mapped onto a standard musical keyboard, enabling experienced keyboard players to learn the system from a standard musical score. We examine the feasibility of the system through a series of evaluations and user studies.

Keywords: live performance, singing voice synthesis, human computer interface, musical instrument, text entry

1. Introduction

The use of a singing voice synthesizer such as VOCALOID [9] has become popular especially in Japan. However, there is little precedent of live improvisational performance using real-time singing voice synthesis even though there is a huge demand for it. This is mainly because it is difficult to input song lyrics at a real-time rate; this is the problem we want to address in this paper.

A possible approach is to use automatic fitting of the predefined original lyrics to the melody currently being played using melody matching. However, this approach has two problems. First, players often modify the melody significantly including addition of grace notes and change of order in a live improvisational performance. Second, the same melodies often appear repeatedly in a song with different lyrics, making it difficult to find the appropriate lyrics from melody alone in improvisational performance.

Another possible approach is to use speech recognition to input lyrics. This allows the user to improvise arbitrary lyrics during performance, but also presents several problems. First, recent popular speech recognition techniques are optimized for recognizing continuous speech as a whole, rather than for recognizing individual characters in a song separately for timed performance. Second, latency is inevitable in speech recognition, but is not acceptable for real-time musical performance. Finally, it is difficult for the player to listen to his or her own performance while vocalizing.

There are a few experimental systems that allow the user to input arbitrary Japanese lyrics during live performance using a combination of vowel and consonant keys [13], [20]. However, they require the user to press two keys simultaneously to input a character, making it difficult to play fast songs.

[1] The University of Tokyo, Bunkyo, Tokyo 113–0033, Japan
[a)] yamamoto-o@is.s.u-tokyo.ac.jp
[b)] takeo@acm.org.jp

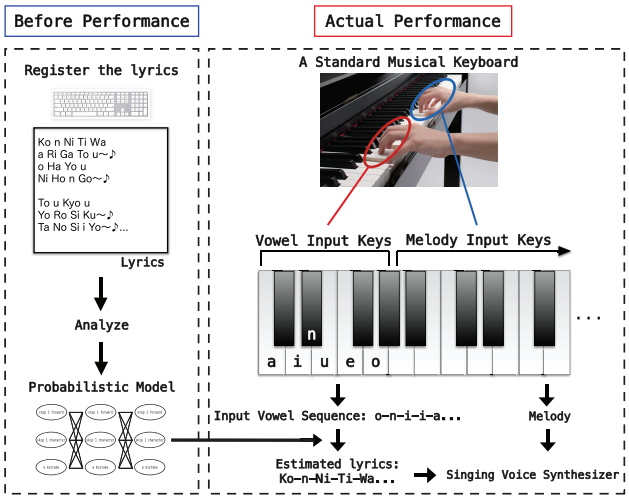


Fig. 1 An overview of the proposed system. Our system consists of two steps, lyric registration step and actual performance step. At the lyrics registration step, the user registers the lyrics of the songs, and the system analyzes it. At the actual performance step, the user simultaneously inputs the vowel sequences and melodies using a musical keyboard, and the system estimates the plausible lyrics from them and synthesizes singing voice sounds.

To address these problems, we propose to use a vowel keyboard to input the lyrics during live improvisational performances (**Fig. 1**: right). In our system, the user inputs the lyrics with one hand using a vowel keyboard and the melodies with the other hand using a musical keyboard simultaneously. Our system allows the user to modify the melodies of a song freely and to pick an arbitrary portion of predefined lyrics during a live performance.

Our system is designed for Japanese lyrics. In Japanese, a character consists of a consonant and a vowel (**Fig. 2**). Hence, multiple Japanese characters match a given vowel. However, we can identify the most plausible character sequence in the predefined lyrics by finding the corresponding vowel sequence using a

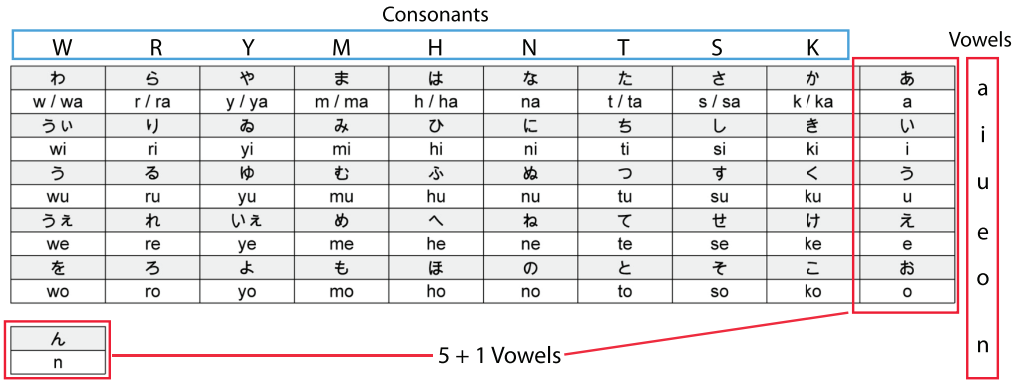


Fig. 2 In Japanese, a character consists of a consonant and a vowel. For example, "Ka" is the combination of "K" and "a," "Su" is the combination of "S" and "u." There are 5 vowels and a special vowel (n) in Japanese.

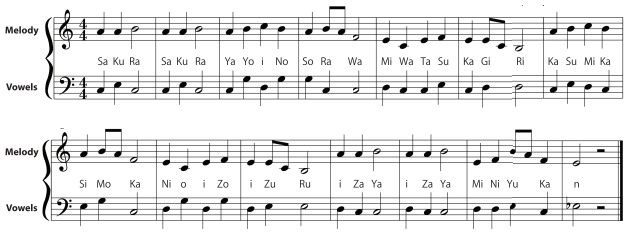


Fig. 3 In our system, the lyrics can be represented as a standard musical score as the left hand part, because vowel keys are mapped onto a standard musical keyboard (Song: SakuraSakura).

probabilistic alignment technique (Fig. 1). Specifically, our system automatically finds a portion of the predefined lyrics whose vowel sequence matches well with the vowel sequence being input by the player. We use a Hidden Markov model for alignment.

There are only five vowels in Japanese, "a," "i," "u," "e," and "o." We also use a special character "n," hence we use six keys to input lyrics. This makes it possible to input vowels rapidly without moving the hand to other locations in contrast to other methods that use many keys to input lyrics. Additionally, by mapping the vowel keyboard onto a traditional musical keyboard, one can represent lyrics as a standard musical score (**Fig. 3**), enabling the user to practice the skills more easily.

One possible criticism is that one can use only the predefined lyrics in our system. It is not possible to compose completely novel lyrics during performance, However, in real improvisational performance, it is actually rare to see the singer composes completely new lyrics during performance. They may improvise novel melodies, but improvised lyrics are usually composed of phrases already in the original lyrics, and this can be handled using our method.

The contributions of this paper are as follows.

(1) Our system allows the user to perform regular Japanese songs at the original tempo, including high-speed songs.
(2) Our system enables the user to rearrange the pieces of predefined lyrics in a live improvisational performance.
(3) We introduce the analogy of a musical score alignment technique to lyrics alignment.
(4) We examine the feasibility of the method with performance evaluation and user studies.

2. Related Work

2.1 Live Performance Using Singing Voice Synthesis

Yamamoto et al. [20] used a combination of a dedicated special keyboard to input lyrics used by the left hand and a standard musical keyboard used by the right hand for improvisational performance (**Fig. 4**). The lyrics keyboard is designed for Japanese, consisting of ten consonant keys and five vowel keys placed to fit the left hand. The user inputs a character by pressing a combination of a consonant key and a vowel key. However, it is difficult for a typical player to press correct multiple keys simultaneously during a live performance.

Formant Bros. [13] assigned lyrics input keys to a common musical keyboard (**Fig. 5**). A character can be input using the triplet three-key combination of a pitch key, a consonant key, and a vowel key. The benefit of this approach is that it enables description of lyrics and melodies as a standard musical score, making the method easy to learn. However, the consecutive triplet chord input is difficult even for professional pianists. Thus, the approach remained at the level of playing slow nursery rhymes, in contrast to regular songs played at a realistic speed (we assume the range of tempo of regular songs is about 50~200 BPM [beats per minute]).

HANAUTAU [17] uses pitch detection from the user's voice inputted by microphone for melody and lyrics typed with both hands using a common QWERTY keyboard. However, using a QWERTY keyboard does not provide input at a speed sufficient to play common music adequately.

A case has been made to use a Flick text input method [12] for live performances using real-time singing voice synthesis to input lyrics. Although the Flick text input method is a fast text entry method, it still can't achieve sufficient input speed for singing a song. Additionally, because it requires two-step control (push and slide), it is difficult to adjust the timing to the music using that method.

DiVA [2] uses CyberGlove and several sensors and measured the hands gestures to control the lyrics. The gestures are trained and trigger a neural network with a given gestural language that associates one posture for each phoneme of English. Reference [6] also measures hands gestures to control both pitch and

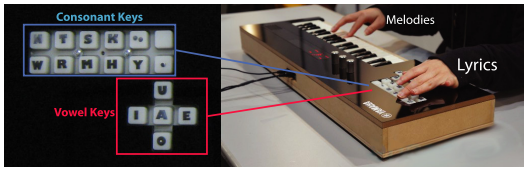


Fig. 4 Yamamoto et al. [18] used the combination of a special lyrics keyboard and a piano keyboard.

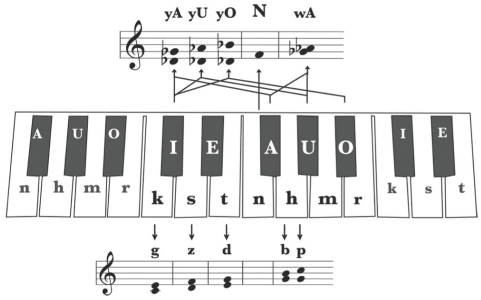


Fig. 5 Formant Bros. [13] mapped the lyrics input keys onto a standard musical keyboard.

the phoneme of singing voice synthesizer. However, the gestural control is difficult for fast songs.

Cantor Digitalis [3] has been used in several musical improvisations using singing voice synthesizer by multi-touch tablet. Their alphabet control is limited in only a few vowels (formants) and can't output the most characters including consonants as a language. Then, their system is inadequate for performing the lyrics of common songs. We address this issue.

2.2 Text Entry

Text input predictions [4], [16] and query word suggestions [8], [22] are not applicable to our target purpose, because they pose two problems for our target. First, there is no way to input the first character of the word the user wants to input at a real-time rate. Second, there is no way to select a candidate in real-time.

Many studies have been conducted regarding word completion [1], [19] from the user's ambiguous input. The word completion methods modify or correct word input by the user including mistypes to form a plausible word. However, these approaches use lazy evaluation. Lazy evaluation estimates the correct word retroactively after the user inputs several words. Thus, it can't be used for our target, which requires outputting the characters individually.

2.3 Musical Score Alignment

Musical score alignment techniques estimate the current playing position of given music (audio or MIDI stream) in a musical score in a database, and use it for various applications such as generating musical accompaniments [15], and displaying the musical score using auto scroll [7], [10].

Recent studies regarding musical score alignment are categorized into two approaches. The first approach is to solve the problem by minimizing some metrics representing how two musical signals differ at each time [21]. This approach is vulnerable to the uncertainty of the user's performance including mistakes, tempo change, or other musical expressions. The second approach is to use probabilistic models [11], [14], [15]. This approach is advantageous because it is robust against such uncertainties in the user's performance. We also use a probabilistic model, but for vowel sequence not for melody.

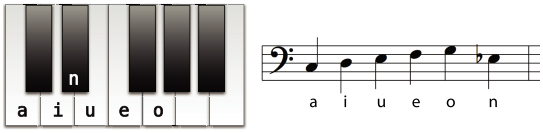


Fig. 6 The vowel input keys are mapped onto a standard musical keyboard. They can be covered by a hand.

3. User Interface

An overview of our system is shown in Fig. 1. Our system requires two steps. The first step is lyrics registration before performance. The second step is actual performance. At the lyrics registration step, the user registers the intended lyrics to be used. The user can register multiple lyrics at a time. At the performance step, the user simultaneously inputs vowel sequences using a vowel keyboard and melodies using a musical keyboard. The system estimates the plausible lyrics from the vowel sequences and synthesizes singing voice sounds. Note that the system does not use a melody sequence for estimation.

The vowel keys are assigned to a portion of a standard musical keyboard. In our prototype, each key mapping is set as "a" to C2, "i" to D2, "u" to E2, "e" to F2, "o" to G2, "n" to E♭2, to fit all the keys in a palm (**Fig. 6**). This makes it possible to represent a vowel sequence as a standard musical score, making it easier for players to practice the performance. In addition, the musical score for our system requires only monophonic phrases of six vowel keys, which is much simpler than that of Formant Bros. Actually, the musical score for our system is much easier than popular piano scores such as J.S. Bach and Chopin.

Our system doesn't require the user to input the vowel sequences strictly in the order of the original lyrics, because the system estimates the probable lyrics using a probabilistic model. This allows the user to jump to arbitrary positions in the lyrics including backtracking. Additionally, our system allows the user to make mistakes, freeing the player from paying excessive attention to vowel input. When the user jumps the position of the lyrics, the system would output a few wrong characters until following the user. However, these wrong characters at least have correct vowels, which limits the level of discomfort.

The timing control is complicated because two keys must be pressed in a coordinated way. Our current implementation is as follows. If a pitch key has already been pressed, the system begins a new voice when a new vowel key is pressed. However, if no pitch key has been pressed, the system does not begin a new voice, when a new vowel key is pressed. If a vowel key has already been pressed, the system begins a new voice, when a new pitch key is pressed. If no vowel key is pressed, the system begins a new voice with the last vowel input by the user, when the user presses a new pitch key. We choose this asymmetric scheme because pitch keys serve as the main control, with vowel keys serving as a modifier. Note that previous systems [13], [20] begin a voice only when 3 keys (vowel, consonant, and pitch) are pressed together, creating difficulty in producing fast real-time

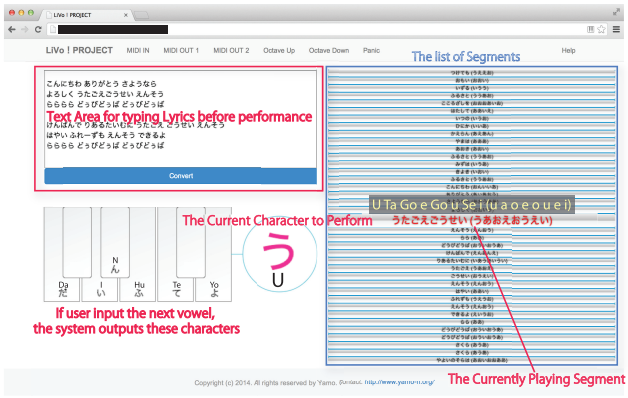


Fig. 7 The user interface view of the LiVo system on a web browser. The user enters the lyrics in the top left text area before performance. The characters displayed on the bottom left keyboard view represent the predicted characters which would be sent to the singing voice synthesizer next when the user would press the vowel key.

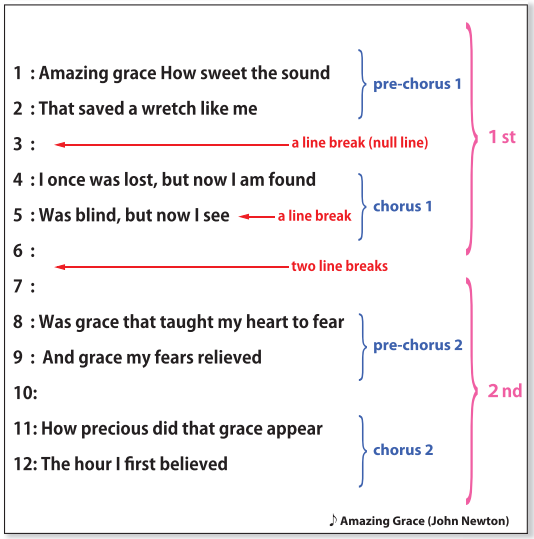


Fig. 8 The text format for registering lyrics before performance. It is required to annotate the rough structures of the song (e.g., verse, chorus, a bunch of musical phrase etc...) by inserting line breaks. We use English here for explanation. The real system only takes Japanese alphabets as input.

performance.

4. Technical Details of Lyric Alignment

This section explains how we estimate the position that the user wants to perform in the lyrics from the vowel sequences input by the user. In the registration step, the system analyzes the lyrics entered by the user, and constructs a data structure to be used in the performance. In the actual performance step, we estimate the most probable lyrics using a Hidden Markov Model (HMM) that encodes the behavior of the movements between consecutive characters in the lyrics and jumps during performance. We search the end point of the Viterbi path in this HMM using multi-agent search to find the best matching lyrics for the given vowel sequence. Note that the estimation solely depends on the vowel input, and does not use melody information at all.

4.1 Lyric Registration Step before Performance

Figure 7 shows the user interface view of the system. The user types the lyrics in the top-left text area and presses the "convert" button to finish. Here, our implementation allows only Hiragana characters as input for simplicity, although Kanji character can be supported. The text format for typing lyrics is shown in **Fig. 8**. The system requires the user to annotate rough structures of the song (e.g., repeating, verse, chorus, or several phrases) manually using line breaks. With more line breaks inserted, the system interprets the point as a larger compartmental boundary. Note that the system doesn't distinguish the kind of a section the user delimited (e.g., it doesn't care about whether a section is verse or chorus), and the user can split the lyrics of the song into arbitrary blocks.

After the user completes text entry, the system decomposes the text into morphemes using morphological analysis, connecting a morphme to the previous morpheme, if the morpheme contains only one character (because we consider it to be unnatural to voice a morpheme consisting of a single character in a song). To be able to execute the morphological analysis, we first machine-translate the Hiragana characters the user inputted to the text including Kanji characters. If the morphological analysis includes errors, the user can fix them manually. Additionally, if a divided character string contains repeating characters such as ("Yu" - "Ra" - "Yu" - "Ra"), the system saves only one instance of the repeat ("Yu" - "Ra") with the number of repeats (in this case, two). We define each delimited character sequence after these operations as a "segment." Each segment becomes the leaf node of the lyrics tree as described in the next paragraph.

Commonly, a song has the hierarchical structure that consists of musical phrases [18]. We construct a hierarchical tree structure (called lyrics tree) of the segments using the user's annotations (**Fig. 9**). The lyrics tree is constructed by dividing the array of segments recursively from a large structure to a small structure according to the annotations (the smallest structure is a segment). The lyrics tree is used for determining the probabilities of each movement in the lyrics as described in the next subsection.

4.2 The User's Behavior Model for Lyric Movements

We model how the user moves from one character to another character in the lyrics during a performance as an Ergodic Hidden Markov Model (HMM) (**Fig. 10**). In our HMM, an unobserved state corresponds to a position in the lyrics and a transition between unobserved states corresponds to a movement in the lyrics. Each transition generates a specific observable symbol deterministically, which is a vowel of a character. For example, if we have a pre-registered lyrics "Ko-n-Ni-Ti-Wa (segment 1), a-Ri-Ga-To-u (segment 2)," the HMM produces an observable symbol "o" when moving to a state "Ko" or "To."

Each state transition probability of our HMM (probability of a movement in the lyrics) is computed according to the kind of movement. For example, moving to the next character is more likely than a jump to a distant location, moving within a segment is more likely than a jump to the different segment, and a jump to the head of a sentence is more likely than a jump to the middle of a sentence. In a similar way, we determined the likelihoods of all kinds of the movement by a heuristic manner. The system

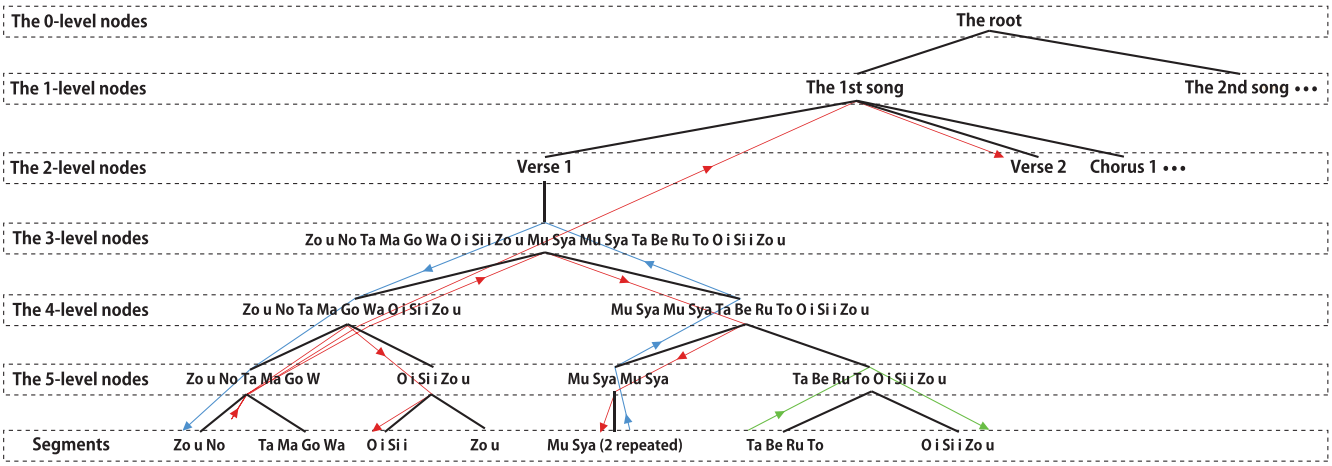


Fig. 9 The hierarchical tree structure of the lyrics segments. The system constructs this structure according to the annotations of the user. The number at the top right of each segment denotes its hierarchical level. The black lines denote the edges. The red, blue and green arrows denote possible movements by the user in this tree.

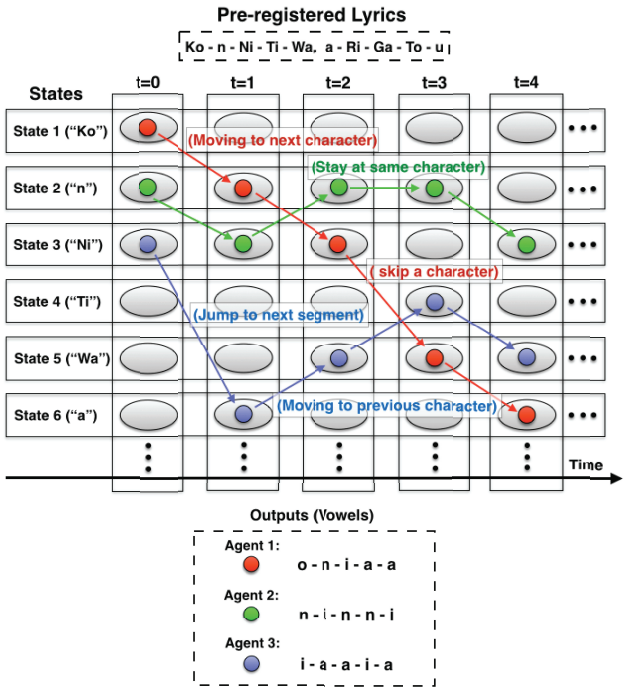


Fig. 10 We model the user's movement between two characters in the lyrics during performance as Hidden Markov Model (HMM). A position in the pre-registered lyrics becomes a state.

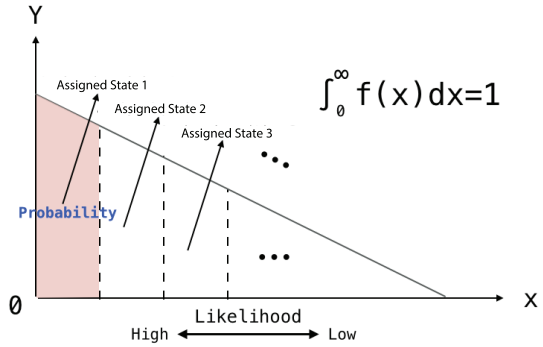


Fig. 11 Each state transition probability of HMM is determined manually by the authors. We prepare a monotonic decreasing distribution function, and assign the divided area to them according to the likelihood of kinds of movement.

first enumerates all the possible movements from the current state and sorts them by the likelihood by the kind of movement. For example, in the above example, if you are at "n," possible destinations are "Ni" (next character), "Ti" (skip a character), "n" (stay at same character), "Ko" (previous character), "a" (jump to the head of the next segment) in the order of likelihood. Next, we define a monotonically decreasing linear function (it fills the condition $\int_0^\infty f(x)dx = 1$) (**Fig. 11**). We split the area surrounded with this function and ($x >= 0$, $y >= 0$ in Fig. 11) evenly along the x-axis, and assign the positions in the sorted list to each split area to be correspond the movement to the position that has higher likelihood to larger area. Finally, the system uses each assigned area (colored area in Fig. 11) as the transition probability to move to the position.

The likelihood of a movement is computed by traversing the lyrics tree. An example is shown in Fig. 9. When the current state is one of the character at a leaf node, it can move to the next character within the same leaf node, or move up to parent nodes and then come down to some other leaf node like red, blue and green arrows in Fig. 9. Each step movement in the tree is associated with a certain cost (e.g., moving to a next character has lower cost than moving to a higher level), and the system accumulates these costs during the traversal. In particular, we multiplied a certain weight ($0.0 < w < 1.0$) according to the traversed number of the tree edges necessary to move to the next position along the tree. We used 0.8 as this weight in our experiment.

4.3 Estimation of the Performed Position in Lyrics during Performance

The most likely position the user wants to perform in the lyrics can be computed as the end point of the Viterbi path that gives the highest accumulated state transition probability (minimum cost) among all the possible paths in the HMM. The Viterbi path is also required to output a vowel sequence that matches with the user inputs. We search this path using a multi-agent search algorithm [5].

The multi-agent search uses multiple interacting intelligent

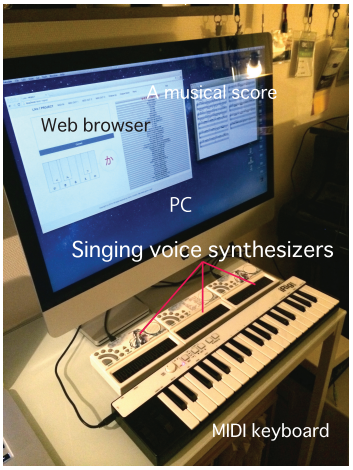


Fig. 12 The system setup.

agents for finding the minimum cost path. Each agent is associated with a state in the HMM (a position in the lyrics), and moves to the next state according to the HMM (color circles in Fig. 10). Since multiple destination states exist for a state, the system generates multiple copies of an agent and associates a copy with each destination state. However, if the movement is an impossible one, the system discards the copy. Each agent is scored by the accumulation of the state transition probabilities between the respective pairs of HMM states passed. For example, the score of the agent 1 in Fig. 10 is determined by using $\pi_0^1 \times \tau_{12} \times \tau_{23} \times \tau_{35} \times \tau_{56}$ where π_0^1 denotes the initial probability of state and τ_{ij} denotes the state transition probability from state i to state j. If the score of an agent becomes less than a threshold, the system destroys the agent. Note that we used 0.01 for this threshold in our experiment. Basically, Lower threshold is better although lower threshold increases the computational cost. When the number of the agents decrease, the system seeds new agents to randomly selected positions and each agent restarts. Additionally, if multiple agents are reached at the same position in the lyrics, the system retains only the agent with the highest score. This procedure corresponds to the pruning process in dynamic programming. Finally, we obtain the optimum position by selecting an agent has the maximum score at the current time. In our experiment, we used 512 agents. Here, we set the initial probabilities of all the states as 1.0. When the system starts, the initial state is randomly selected, or the user selects manually on the right pane of the UI view (Fig. 7).

5. Evaluation

5.1 Implementation

Figure 12 shows the system setup for LiVo. The software for the system is written in Javascript, runs on web browser. We used YAMAHA NSX1-board for a singing voice synthesizer. The MIDI keyboard and the singing voice synthesizer are connected to the computer by USB, and the web browser communicates with them by Web MIDI API.

We measured the computation time from receiving a NOTE ON message from MIDI keyboard to sending a NOTE ON messages to the synthesizer via MIDI OUT. Our equipments are CPU: 2.6 GHz Intel Core i7, RAM: 16 GB, Web Browser: Google Chrome. This latency was less than 1 [ms]. Thus, it is negligible for human perception. In our implementation, total memory requirement for the system was about 44 [MB] with 1,000 characters lyric.

Table 1 The song list used for experiments. The "following" column represents the numbers of output wrong characters after jumping the positions in the second experiment in Section 5.2. These numbers represent higher abilities as lower number.

Title	Author	Characters	Following
1. Senbon Sakura	KurousaP	616	3
2. World is Mine	supercell	628	2
3. Melt	supercell	432	2
4. HatsuneMiku no Shoushitsu	BousouP	1373	4

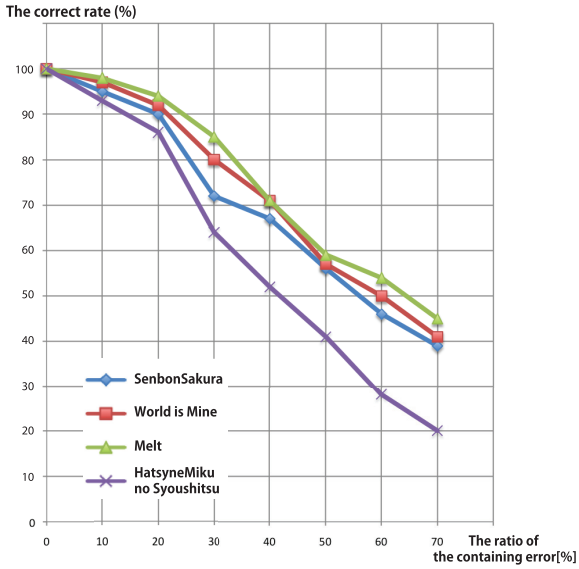


Fig. 13 The robustness against mistakes during performance. The horizontal axis: The ratio[%] of numbers of the inserted error vowels to the total numbers of characters in the song. The vertical axis: The correct rate [0,100]. Higher value represents higher accuracy.

5.2 Accuracy of the Alignment Algorithm

First, we examined robustness against mistakes. **Table 1** shows the song list used for our experiments. The original vowel sequences are defined as the vowel sequences extracted from the lyrics of the original songs. We generated random error vowels by the uniform sampling from six vowels ("a," "i," "u," "e," "o," and "n"), and inserted them into random positions between the two vowels of the original vowel sequences. The inserted positions are selected randomly according to an uniform distribution ranging between one and the total number of characters in the song minus one. Note that we prohibited inserting more than three error vowels sequentially. We inputted this vowel sequences containing error vowels into our system as a source and examined the error rate of the output characters compared to the original lyrics.

The relationships between the numbers of inserted errors and the output error rate are shown in **Fig. 13**. Although over 30% errors cause a fatal decrease in accuracy, the system maintains more than 85% accuracy for 10~20% errors. These results show that the system is reasonably robust against errors in vowel input.

In a second experiment, we examined robustness against jumps to irregular destinations in the lyrics. The songs used for this

Fig. 14 The performance scene by a professional pianist.

Fig. 15 Two scenes at the workshop.

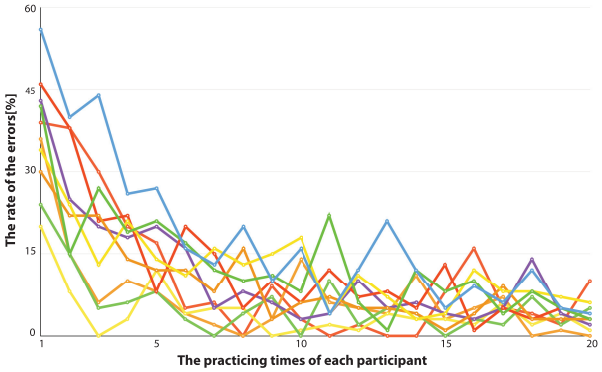


Fig. 16 The amount of vowel input mistakes made by test participants plotted against the number of practices. The vertical axis shows the ratio of the errors to the total numbers of characters in the song and the horizontal axis shows the number of practice.

experiment are the same as those used for the previous experiment. We randomly rearranged the lines in the registered lyrics (this lines correspond to the rows in Fig. 8), and used their vowel sequences as input. When the input vowel sequence moves to the next randomly rearranged line, the system first makes several errors because it expects the next line in the original lyrics. However, the system eventually identifies the correct line after observing several incoming vowel keys. We counted the maximum number of incorrect output characters before identifying the correct line.

The result of this experiment is shown in Table 1 as "following" column. The result shows that robustness against jumps to irregular destinations is strongly affected by the total number of characters in the lyrics. However, even for "HatsuneMiku no Shoushitsu", which has the most characters in our experiments, our system successfully finds the correct segments within a few characters, which is mostly shorter than a single morpheme in Japanese. This result shows that the system is reasonably robust against improvisational jumps during performance.

5.3 Playability

We examined the playability of our system by a professional pianist (**Fig. 14**). We selected a famous Japanese song "Senbon-Sakura" (Author: KurousaP) for performance. This song is one of the Japanese songs with the fastest tempo (BPM: 152), and contains fast movements (including a lot of sequential 16 parts of note phrase) between the characters in the lyrics. We picked this song as a stress test for the system. The time for practice was one week, one hour per day.

As the result, this song can be performed at the original tempo at the performance. The total length of the performance was four minutes, five seconds. The pianist adds several musical expressions including ad-lib modification of the melody. These expressions can't be performed using any existing methods. Additionally, the system outputted plausible lyrics, even though the pianist often made mistakes and improvisational changes.

In addition, we recruited five participants and played the recorded sound of this performance for them. We asked them to report the number of times they felt an unnatural singing voice sound. No one reported this more than three times.

Furthermore, some users including the pianist of this experiment and authors (7 participants) have tried to play Formant.Bros' system [13] and Yamamoto et al.'s system [20] for the comparisons of the playability. However, no one was able to play the song used in our experiment with their systems at the same tempo. This shows that it is almost impossible to play such a fast song with existing systems.

5.4 Workshop

We held a workshop with ten amateur pianists, all of whom have played with piano or other musical keyboards for more than five years. We gave the players a musical score for our system and asked them to practice. We selected a standard jazz song "Autumn Leaves" (Music: Joseph Kosma, Lyrics: Junko Akiyama) with the tempo 120 [BPM] for practice. All participants had already known this song before the workshop. We printed the score at this study. The user checked the vocal singing sound only by hearing. The time to practice was three days, 30 minutes per day for all participants (**Fig. 15**).

We monitored the ratio of the number of vowel input mistakes to the total number of characters of the song in the case of playing the musical scores strictly by repeating practices. The result is shown as **Fig. 16**. This figure shows the relationships between the ratio of the different played notes to the original and practice times of each participant. We found that the ratio decreases rapidly with repeating practice. Then, we verified that our system doesn't require any further skills for common pianists.

After the practice, we requested the participants to play the song to an accompaniment for more than six choruses repeatedly, while modifying the melodies and jumping freely ad-lib. As described in §Introduction, most existing approaches, such as melody fitting, can't be used for this scenario. We included this actual scene in the supplemental video. The video shows each pianist playing the song while modifying the melody significantly ad-lib. The participants re-mixed the pieces of the predefined

lyrics flexibly. This new type of musical expression is enabled by our system. Note that all participants we employed already had a skill for improvising music by piano before the experiment. So, special trainings for improvisation were not required, even including the control of lyrics.

After the workshop, We conducted individual interviews. At the interviews, we asked three questions. First, about the difficulty of the system: easy, neutral, difficult, impossible. If difficult or impossible, we were going to ask the reason. But, all participants answered it's easy. Second, about the playability. We asked whether the user has any dissatisfactions for expressing the musical phrases they want. Finally, we requested free comments.

One participant mentioned that our system outputs the correct vowel component even if the system outputs wrong lyrics after a jump. She said this property is advantageous because it can minimize the sense of discomfort in the hearer.

Some participants (6/10) mentioned that they feel discomfort at first because they already have substantial experience playing piano and can read musical scores, but the vowel keys we assigned on the musical keyboard don't correspond to the heard pitches of the synthesized sound. However, they also mentioned that the sense of discomfort decreased gradually as they adjusted to our system.

Nearly all participants (8/10) said that the system allow input mistakes to some extent and it was a nice feature. They added that practice for our system was substantially easier than they expected, because the lyrics are represented as a standard musical score. We consider these to be significant merits of our system.

Others (3/10) said that they often worried about the latency of the "S" consonant when voicing. The characters that have the "S" consonant are sibilant. Sibilant characters are voiced after a comparatively long breath sound, and the timing of voicing the vowel is perceived as the rhythmic timing. Then, if the user presses a key to start a voicing sibilant on the exactly the timing in the rhythm, the sound will be perceived late. In the case of voicing by mouth, we usually overcome this problem through practice, intentionally beginning to voice a character with the "S" consonant slightly earlier. However, the participants had not mastered it for our system in our limited study. We think that this requires substantially more training.

6. Conclusion and Future Work

In this paper, we proposed a practical user interface that enables the use of real-time singing voice synthesizer at an improvisational live performance by inputting the lyrics and melodies of songs simultaneously using a standard musical keyboard. The proposed system allows arbitrary movements within the lyrics including jumping, backtracking, and mistakes by estimating plausible lyrics from vowel sequences using a probabilistic model. Additionally, the lyrics for our system can be described using a standard musical score making it easier to learn. As a result, we achieved flexible control of lyrics and melody using our system at real-time rate that enabled live improvisational performance of distinctive musical expressions.

However, our study leaves several problems to be addressed in the future. First, our approach is dependent on the specific properties of the Japanese language, which is a mora language with only five vowels. Consequently, it is difficult to apply to other languages such as English. Second, in our system the interpretation of the structure of a song depends on manual annotations by the user. We expect that it will be convenient to have the system automatically extract this structure by using natural language processing techniques. In addition, our method has many parameters, such as the state transition probabilities in the HMM, determined empirically by the authors. Automatic learning and adjustment of these parameters is an important task left for future work.

References

[1] Buschek, D., Schoenleben, O. and Oulasvirta, A.: Improving Accuracy in Back-of-Device Multitouch Typing: A Clustering-based Approach to Keyboard Updating, *IUI'14*, pp.57–66, ACM Press (2014).

[2] d'Alessandro, N., Wang, J., Pritchard, R. and Fels, S.: Bringing Bio-Mechanical Modelling of the OPAL Complex as a Mapping Layer for Performative Voice Synthesis, *9th International Seminar on Speech Production* (*ISSP*), pp.111–118 (2011).

[3] Feugere, L., d'Alessandro, C. and Doval, B.: Performative voice synthesis for edutainment in acoustic phonetics and singing: A case study using the Cantor Digitalis, *5th International ICST Conference*, pp.169–178 (2013).

[4] Gargi, U. and Gossweiler, R.: QuickSuggest: Character Prediction for Improved Text Entry on Web Appliances, *International Conference on the World Wide Web*, pp.1249–1252 (2010).

[5] Goto, M. and Muraoka, Y.: Beat Tracking based on Multiple-agent Architecture — A Real-time Beat Tracking System for Audio Signals, *2nd International Conference on Multiagent Systems*, pp.103–110 (1996).

[6] Ito, M.: A singing voice synthesizer controlled by arm motions using compressed phoneme determination algorithm, *Proc. IIH-MSP* (2014).

[7] Joder, C., Essid, S. and Richard, G.: A Conditional Random Field Framework for Robust and Scalable Audio-to-Score Matching, *IEEE TASLP*, Vol.19, No.8, pp.2385–2397 (2011).

[8] Kamvar, M. and Baluja, S.: Query Suggestions for Mobile Search: Understanding Usage Patterns, *ACM Computer Human Interaction* (*CHI*), pp.1013–1016 (2008).

[9] Kenmochi, H., Ohshita, H. and YAMAHA Corporation: VOCALOID — Commercial Singing Synthesizer Based on Sample Concatenation, *INTERSPEECH*, pp.4009–4010 (2007).

[10] Lemouton, S. and Schwarz, D.: Score Following: State of the Art and New Developments, *New Interfaces for Musical Expression* (*NIME*), pp.36–41 (2003).

[11] Maezawa, A., Okuno, H.G., Ogata, T. and Goto, M.: Polyphonic Audio-to-Score Alignment based on Bayesian Latent Harmonic Allocation Hidden Markov Model, *International Conference on Acoustics, Speech and Signal Processing* (*ICASSP*), pp.185–188 (2011).

[12] Mikumin, P.: Flick input for realtime singing synthesizer (2013), available from ⟨http://www.nicovideo.jp/watch/sm17357529⟩.

[13] Miwa, M. and Sakonda, N.: BROTHERS' Button to Phoneme Transfer Standard for International Language, *Departmental Bulletin Paper for Nagoya University of Arts and Science*, Vol.6, pp.21–33 (2013).

[14] Nakamura, E., Nakamura, T., Saito, Y., Ono, N. and Sagayama, S.: Outer-Product Hidden Markov Model and Polyphonic MIDI Score Following, *Journal of New Music Research*, Vol.43, No.2, pp.183–201 (2014).

[15] Nakamura, T., Nakamura, E. and Sagayama, S.: Acoustic Score Following to Musical Performance with Errors and Arbitrary Repeats and Skips for Automatic Accompaniment, *Sound and Music Computing Conference* (*SMC*), pp.200–304 (2013).

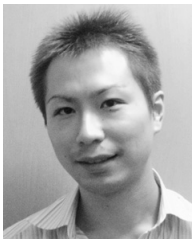
[16] Silfverberg, M., MacKenzie, S. and Korhonen, P.: Predicting Text Entry Speed on Mobile Phones, *The ACM Conference on Human Factors in Computing Systems* (*CHI*), pp.9–16 (2000).

[17] Takemoto, T., Baba, T. and Katayori, H.: A Real-Time Singing Generator Using Typing and Humming "Hanautau" Based on an Agile Software Development, *Interaction, Information Processing Society of Japan* (2014).

[18] Watanabe, K., Matsubayashi, Y., Inui, K. and Goto, M.: Automatic Japanese Lyric Generation Based on The Hierarchical sSructure of The Song, *The Association for Natural Language Processing*, pp.694–697 (2014).

[19] Wobbrock, L. and Wigdor, J.O.: Typing on Flat Glass: Examining

Ten-Finger Expert Typing Patterns on Touch Surfaces, *ACM Computer Human Interaction* (*CHI*), pp.2453–2462 (2011).
[20] Yamamoto, K., Kagami, S., Hamano, K. and Kashiwase, K.: The Development of a Text Input Interface for Realtime Japanese Vocal Keyboard, *Journal of Information Processing*, Vol.21, No.2, pp.274–282 (2013).
[21] Yamamoto, R., Sako, S. and Kitamura, T.: Robust on-line Algorithm for Realtime Audio-to-Score Alignment Based on a Delayed Decision and Anticipation Framework, *International Conference on Acoustics, Speech and Signal Processing* (*ICASSP*), pp.191–195 (2013).
[22] Zhang, X., Zilles, S. and Holte, R.C.: Improved Query Suggestion by Query Search, *35th German Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence*, Vol.7526, pp.205–216 (2012).



Kazuhiko Yamamoto was born in 1985. He recieved his Master of Design degree from Kyushu Univeresity in 2010, and has been engaged in YAMAHA corporation since 2010. He also joined the University of Tokyo as a Ph.D. student in 2013. His research interests include computer graphics, numerical simulation, human computer interface, and audio/image signal processing.

Takeo Igarashi received his Ph.D. from the Department of Information Engineering at The University of Tokyo in 2000. He joined the University of Tokyo as an Assistant Professor in 2002, and became a Professor in 2011. He also served as a director for JST ERATO (2007–2013). His research interest is in user interfaces and interactive computer graphics.