LiVo:リアルタイム歌声合成演奏インタフェース

山本 和彦　　五十嵐 健夫 *

概要. 本稿では, リアルタイムに歌声合成を使った音楽演奏表現を可能にするための歌詞入力インタフェースを提案する. 提案手法では, 鍵盤の一部に日本語の母音を割り当てて, ユーザに歌詞文字列の母音部分を抽出したものを入力させ, そこから最も尤もらしいデータベース中の元の歌詞を推定して歌声合成する. 日本語では母音は 5 つしかないため, 片手で非常に高速に入力できる. また, 一般的な鍵盤に割り当てられているため楽譜で表現して容易に練習可能である. さらに, ユーザによる歌詞中の移動を確率的生成モデルで表現しているため, 後戻りを含めたジャンプや入力ミスにも柔軟に対応できる.

1 はじめに

VOCALOID [4] や UTAU [3] 等の歌声合成ソフトウェアが普及し, 歌声合成をメインボーカルとして利用した音楽制作は一般的なものとなった. しかし, リアルタイムに歌声合成を使ったライブパフォーマンスは, 強い要望があるにも関わらず限られた例があるのみである. これは主に, 歌詞を実用的な速度で入力する方法が存在していなかったことに起因する. 本稿では, この問題を解決し, 歌声合成による新たな音楽表現の技法を開拓することを目的とする.

2 関連研究

山本ら [6] は左手の親指で母音ボタン, その他の指で子音ボタンを同時に押して, その組み合わせでリアルタイムに文字入力して歌声合成演奏するインタフェースを提案した. しかし, 複数のボタンの組み合わせを実用的な速度で入力するのは非常に困難であるという問題がある.

フォルマント兄弟 [1] は, 母音と子音を通常の鍵盤に割り当てて, 音高と合わせて 3 和音の組み合わせによって歌声合成を使った演奏を試みた. この手法には既に世に広く普及している五線譜として文字を表現でき, 練習が容易になるというメリットがある. しかし, 連続した 3 和音のフレーズは熟練したピアニストにおいても非常に困難であり, 実用的な速度で入力することはできない.

HANAUTAU [2] はマイク入力で声からのピッチ抽出を行うことで両手を自由にし, QWERTY キーボードでユーザは歌詞を入力する. しかしタイピングによる文字入力速度は音楽演奏に対しては決して十分なものとは言えなかった.

* 東京大学

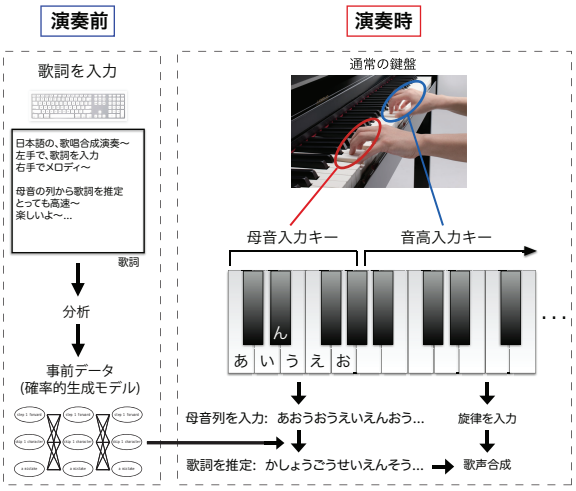

図 1. LiVo システム概要

3 ユーザインタフェース

図 1 がシステムの全体像である. 鍵盤の一部に日本語の母音 (「あいうえお」 + 「ん」) が割り当てられている. ユーザは歌詞の母音部分のみを抽出したもの (例えば「こんにちわ」ならば「おんいいあ」) を左手で入力する. また同時に右手で残りの鍵盤を使って音高を入力する. システムは事前情報として与えられていたデータベースを使って, 母音列からユーザの意図した元の歌詞を推定し, それを使って歌声合成する.

合成音の発音タイミングは以下の通りである. まず, 音高キーのみを押すと, 最後に出力されていた文字で発音する. 歌詞キーのみを押しても文字の更新は行われるが発音はしない. 歌詞キーを押したまま音高キーを押すと現在推定されている文字で発音する. 音高キーを押したまま歌詞キーを新たに入力すると新しく推定された文字で発音し直される.

演奏に使う歌詞は, 演奏前にユーザに事前入力させる. 入力フォーマットとして, A メロやサビ, 音楽

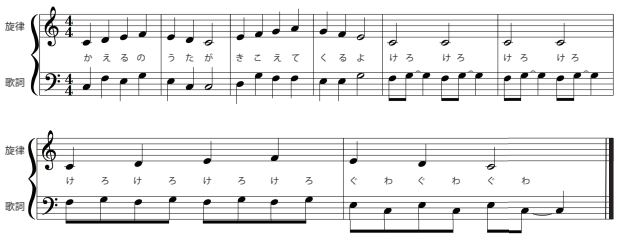

図 2. 旋律と歌詞の楽譜

的なフレーズといった大まかな区切りごとに改行でアノテーションする必要がある. 改行が多く挿入される程, システムはその部分を音楽的に大きな区切りだと判断する.

母音キーは鍵盤に割り当てられているため, LiVo システムは五線譜を使って容易に練習することができる (図 2). 歌詞の入力に母音のみの単旋律しか要求しないため, フォルマント兄弟の手法と比較しても非常に簡単な楽譜で表現することができる.

4 歌詞アライメント

提案手法は, 確率的生成モデルを使った楽曲アライメント [5] のアナロジーを歌詞アライメントに導入したものである. 楽曲アライメントでは, 与えられた音楽の現在の演奏位置が, 事前情報として与えられていた音楽や楽譜のどこの位置に当たるかを推定する. 一方提案手法では, 演奏されている母音列から本来の歌詞中の位置を推定する.

まず, ユーザによる歌詞中の移動の仕方 (一文字進む, 一文字飛ばす, 入力ミスする, 一文字後戻りする, 等) を状態として持つ Hidden Markov Model (HMM) を定義する. この HMM はユーザが母音を入力する度に状態遷移する. さらに同時に, 遷移した状態の指し示す歌詞位置移動方法に従ってシステムは歌詞中を走査する. HMM は, 結果としてたどり着いた位置の歌詞の母音を確率 1 で出力する. このとき, ユーザが演奏している最も尤もらしい歌詞中の位置は, HMM の Viterbi 経路の終点となる. システムはこの Viterbi 経路をユーザが新たな母音を入力する度に計算し直し, 終点の歌詞を出力する.

実際の HMM の構築のためには, 状態の種類の定義とそれらの間の遷移確率が必要である. まず, 経験則に基づいて, 考えられ得る限りの移動方法を優先度が高いと思われる順に羅列する. これら一つ一つを HMM の状態とすると, それらの間の遷移確率 τ_{ij} (状態 i から状態 j への遷移確率. 添字が低い程優先度が高い) の満たすべき条件は, $\sum_j \tau_{ij} = 1$, $\tau_{ii} \geq \tau_{jj}$ $(i < j)$, $\tau_{ij} \geq \tau_{ik}$ $(j < k)$ となる. この条件を満たすようそれぞれの遷移確率を決定する. 結果的に, この HMM の遷移確率は遷移先の状態にのみ依存することになる.

5 ユーザスタディ

アマチュアピアニスト 10 人を集め, 一日 30 分× 3 日間, システムを練習させた. 使用した楽曲は「枯れ葉」(作曲:Joseph Kosma, 作詞:秋山順子) である. このときの練習を重ねるごとの, 歌詞文字総数に対する母音入力ミス率を測定した結果, ほぼ全てのユーザにおいて, 数回練習を繰り返すのみで急激にミス率は低下した. このことからも提案手法は一般的なピアニストにとって決して難解なものではないことが分かる.

さらに練習後, 歌詞の様々な部分を繋ぎ合わせてアドリブ演奏させた. メロディはオリジナルのメロディを弾くことを禁止し, その場で伴奏に合わせて新しく創出してもらった. 結果として, ユーザは既存の歌詞の断片を自由にリミックスして新しい歌詞とメロディをリアルタイムに創出するという全く新しい音楽表現を行った.

6 まとめ

本稿では, 母音列に対して事前情報として与えた歌詞の位置をアライメントすることで, リアルタイムに片手で実用的な歌詞操作を実現することのできるインタフェースを提案した. 演奏時の歌詞中の文字列進行は HMM でモデル化され, 入力ミスや任意の文字列の繰り返し, 大きな演奏位置移動に対しても堅牢に動作する.

参考文献

[1] Miwa, M., Sakonda, N. BROTHERS' Button to Phoneme Transfer Standard for International Language, Departmental Bulletin Paper for Nagoya University of Arts and Science, Vol.6, p21-33, 2013.

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

[3] UTAU, http://utau2008.web.fc2.com.

[4] Kenmochi, H., Ohshita, H., YAMAHA Corporation. VOCALOID - Commercial Singing Synthesizer Based on Sample Concatenation, INTERSPEECH, p4009-4010, 2007.

[5] Maezawa, A., Okuno, H. G., Ogata, T., 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), p185-188, 2011.

[6] Yamamoto, K., Kagami, S., Hamano, K., Kashiwase, K. The Development of a Text Input Interface for Realtime Japanese Vocal Keyboard, Journal of Information Processing, Vol.21, No.2, p274-282, 2013.