社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－HCI－128 (1)
2008－MUS－ 75 (1)
2008／5／28

データフロー図に基づく リアルタイム映像合成システム

小林 敦友 †， 志築 文太郎 †， 田中 二郎 †

† 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻

音楽イベントなどにおいて行われるライブ映像パフォーマンスでは、演者はその場の雰囲気に応じて映像を合成し切り替えていく。それゆえ、観客に映像を提示している間に、次に提示する映像のための映像素材選別やエフェクト適用、合成結果の確認等を行う。本システムでは、映像処理の流れをデータフロー図を使って表現し、映像再生中にもその図を複数個編集可能にすることで、提示中の映像とは別の映像を容易に合成し提示することを可能にした。

Realtime Video Composition System Based on Dataflow Diagram

Atsutomo KOBAYASHI † Buntarou SHIZUKI † Jiro TANAKA †

† Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba

On live video performance, such as in musical events, performers composite and switch videos according to the atmosphere and the occasion. Therefore, the performers select video materials, apply effects and check the composition result while showing video to a audience. Our system express video processing flow with dataflow diagram and allow the performers to edit the diagrams even while a video is running so that the performers composite and show the other videos from the showing one.

1 はじめに

音楽イベントやファッションショーなどにおいて行われるライブ映像パフォーマンスでは、演者はその場の雰囲気に応じて映像を合成し切り替えていく。それゆえ、観客に映像を提示している間に、次に提示する映像のための映像素材選別やエフェクト適用、合成結果のプレビュー等を行う。このようなライブ映像パフォーマンスのためのシステムをソフトウェアで実現するには既存の映像編集、オーサリング用のアプリケーションとは異なるユーザインタフェースが必要となる。我々は、ライブ映像パフォーマンスに適したアプリケーションのユーザインタフェースの設計と実装を目的に、映像合成システムの開発を行ってきた[8]。

1.1 ライブ映像パフォーマンスにおける作業

まず、背景としてライブ映像パフォーマンスで行われる作業について述べる。本研究で対象とするライブ映像パフォーマンスにおける作業手順は、Lewによる3段階のDJ(Disc Jockey)の作業手順[6]と同等のものを想定する。その3段階のDJの作業手順を以下に示す。

1. **メディア検索**：レコードを選択する。
2. **プレビューと調整**：ヘッドホンでレコードをプレビューし、目的の曲やサンプルを見つけ、再生速度、フィルタ、エフェクトの調整を行う。
3. **ライブ操作**：2で調整された素材を、再生されている既存のストリームに組み入れ、スクラッチ、カット、逆再生によってその素材を操作する。

DJ パフォーマンスにおいて音楽が途切れず、音楽の切り替わりが自然であることが要求されることと同様に、ライブ映像パフォーマンスにおいては映像が途切れないことと、映像の切り替わりが自然であることが要求される。つまり上記の手順は、レコードを映像素材に、ヘッドホンをモニタ出力に置き換えるとライブ映像パフォーマンスの手順としても利用できる。事実、motion dive (株式会社デジタルステージ) や V-4(ローランド株式会社) など、この手順を想定した多くのシステムや機材が存在する 。この手順を実現するため、演者はビデオミキサを中心としたシステム構成を使う（図 1)。ビデオミキサは観客に提示するためのメイン出力とプレビュー用のモニタ出力を備えている。演者は、ある映像がメイン出力にて観客に提示されている間に、次に提示する映像素材選別やその映像素材に対するエフェクト適用、調整をモニタ出力でプレビューしながら行う。その後映像を切り替えたり、エフェクトのパラメータを操作する。これらを繰り返すことによって、その場で次々と映像を切り替えていく。このことから、ライブ映像パフォーマンスには、映像素材やそれにエフェクトを加えた映像経路が複数、それら複数の映像経路を切り替るビデオミキサ、及びプレビュー用のモニタ出力が必要であることがわかる。

1.2 ビデオミキサを使ったシステム構成の問題点

上で述べたビデオミキサを使ったシステム構成の問題点として以下の二つがあげられる。

一つ目はライブ映像パフォーマンス中にシステム構成を変更することができないことである。例えば図 1 に示したシステム構成を使ってライブ映像パフォーマンスを行っている最中には、ビデオデッキからの映像にエフェクタ A を適用するように配線を繋ぎ換えることは出来ない。そのような繋ぎ換えを行うにはメインアウトプットへの出力を一旦止める必要があるからである。これは演者がパフォーマンス中に手を加える余地を少なくしており、ライブ映像パフォーマンスに求められる即興性を制限している。

二つ目は、多数のパラメータを変化させることが困難な点である。演者はエフェクタやビデオミキサに備え付けられたつまみやスライダなどを手で操作してパラメータを変化させる。これは直感的に操作できる一方、同時に変化させられるパラメータの数

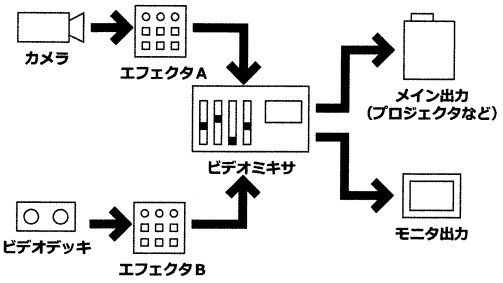

図 1　ライブ映像パフォーマンスのシステム構成例

は限られてしまう。映像表現として時間的変化は重要であり、この問題を解決することは表現の幅をより広げることに繋がる。

1.3 本研究でのアプローチ

上で述べた二つの問題点を解決するために我々のとったアプローチは、上で述べた作業手順の「2. プレビューと調整」で演者が出来ることを増やすことにより、即興性や表現の幅を向上しようとするアプローチである。

一つ目の問題点に対しては、DF（データフロー）図によって映像処理の流れを表し、その DF 図の編集を映像再生中であっても行えるようにした。これにより、演者は任意のシステム構成を構築でき、パフォーマンス中であってもそのシステム構成を変更できる。これは上で述べた作業手順の「2. プレビューと調整」の手順を対象としている。

また、二つ目の問題点に対しては、キーフレームアニメーションを導入し、タイムラインノードと呼ぶ DF 図上のノードとして表現することによって解決を図った。これは上で述べた作業手順の「3. ライブ操作」での作業の一部を「2. プレビューと調整」の手順で済ませてしまおうとするものである。

2 関連研究

映像を扱う DF 型のシステムとしては Jitter(Cycling'74) が挙げられる。Max/MSP(Cycling'74) のプラグインで映像を扱えるもので、ライブ映像パフォーマンスで使われる例もあるが、再生中（実行中）の DF 構造の変更は想定されておらず、あらかじめ作成したプログラムを使ったパフォーマンスしかできない。また、ビジュアルプログラミング言語（VPL）環境として作られており自由度の高いプログラミングが可能な

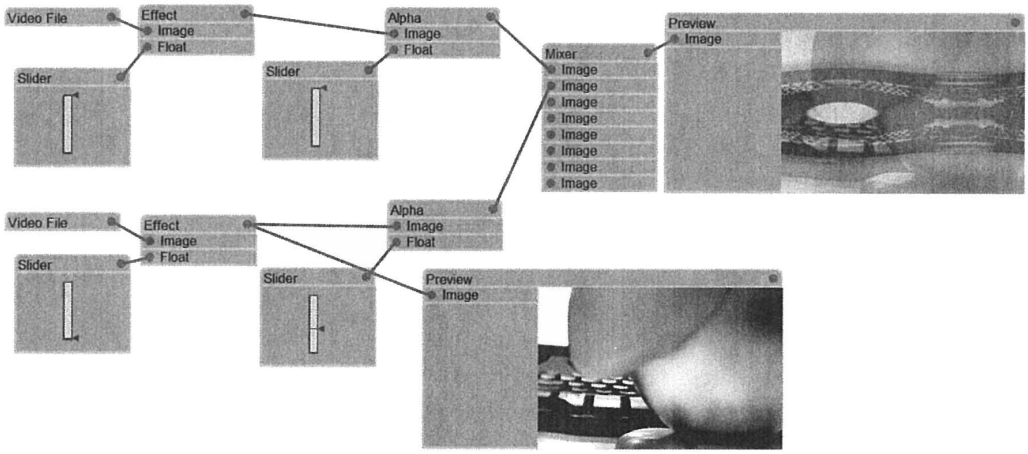

図 2　開発したシステム（図 1 と同様の構成）

反面、ユーザには高度な知識を要求するという点でも本システムとは異なる。

ライブ映像パフォーマンス用のシステムに計算機を使おうという試みとしては、福地らの EffecTV[3] がある。これはコンピュータをエフェクタとして使おうというものであるが、一台のコンピュータを一台のエフェクタとして使うのでそのシステム構成はハードウェアによる結線を前提としている。また、幾つかの商用ソフトウェアも発表されており、代表的なものとして、motion dive があるが、これらのシステム構成は固定されておりエンドユーザが自由に変更できるようにはなっていない。

ライブ映像パフォーマンス用の操作インタフェースとして、徳久らによる Rhythmism[2] や Nervixxx[7]、Bongers らによる Video-Organ[1] などがあげられる。これらは、上で述べた、3 段階の DJ の作業手順での「3. ライブ操作」を対象としており、「2. プレビューと調整」を対象とした本システムとは異なる。しかし、本システムで「2. プレビューと調整」を行い、これらの操作インタフェースによって「3. ライブ操作」を行うといった共存も可能である。

3　DF 図に基づくリアルタイム映像合成

本研究ではライブ映像パフォーマンスで求められる要件を満たしつつ、上で述べた問題点を解決するために、DF 図によって映像処理の流れを表すソフトウェアを開発した。開発したシステムを図 2 に示す。

DF 図とはデータの処理と流れを表す図であり、データの処理をノード、データの流れをエッジとした有向グラフとして表わされる。特に信号処理用の VPL において広く使われている[5]。DF 図を用いた VPL の例としては、音楽、音響用 VPL の Max/MSP や可視化用 VPL の AVS/Express(Advanced Visual Systems Inc.) などが挙げられる。また、プログラミング以外の目的で DF 図が用いられる例としては、飯崎らによる FindFlow[4] などが挙げられる。

本システムにおいてエッジは、映像データと、エフェクトなどのパラメータのための実数データの流れを表現している。ノードは、映像ファイル再生、実数値を指定するスライダー、出力のための映像表示、各種エフェクタやミキサなどを表現する。本システムで対象としているユーザはライブ映像パフォーマンスを行うアーティストであり、映像機材とその接続に精通しているので、映像機材をノードとした DF 図を容易に理解できることが予想できる。例えば、図 1 で示した構成例はある種の DF 図であり、この構成を設置できるユーザは図 1 を容易に理解することが期待できる。そのため、各ノードがそれぞれ映像機材をひとつづつ表現するようになるべく配慮し、本システムの DF 図がユーザにとって理解しやすいことを目指した。ユーザはこの DF 図を編集することによって任意のシステム構成を構築する。DF 図は有向グラフの構造を表現でき

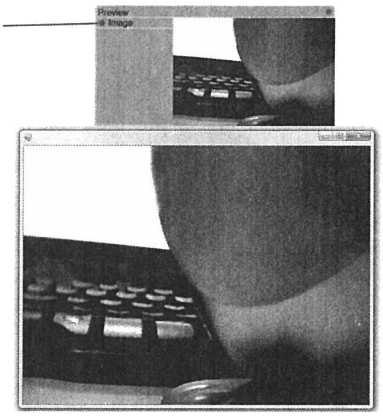

図 3　映像表示用のノード（上）とそのノードをクリックすると開かれる出力ウィンドウ（下）

るので、複数の映像経路を持った構成を表現することができる。これに加え、本システムでは映像再生中であっても DF 図を編集できるようにすることで、パフォーマンス中でもシステム構成を変更できるようにした。これにより、既存のシステムにはできなかった、即興性の高いパフォーマンスが可能になる。

映像表示用のノードは DF 図上のそのノード自身に映像を表示する。さらにその表示部分をクリックすることで、別のウィンドウで出力ウィンドウを開きその出力ウィンドウにも映像を表示する (図 3)。この出力ウィンドウを複数の映像出力を持ったグラフィックカードの一つの映像出力にフルスクリーンで表示することにより、観客への提示用メイン出力として使える。また、ユーザはこの映像表示用のノードをいくつでも作ることができるので、メイン出力以外のものをプレビュー用モニタ出力として使うことができる。

3.1　タイムラインノード

既存の映像合成システムにおいてよく使われる、パラメータの時間的変化手法としてキーフレームアニメーションがあげられる。その表現手法としては横軸に時間軸をとり縦軸に変化させるパラメータをとったタイムラインで表現されることが多い。このタイムラインを複数重ねることで複数のパラメータ変化の指定が可能である。例として Adobe After Effects(Adobe Systems Incorporated) のタイムライン部分を図 4 に示す。

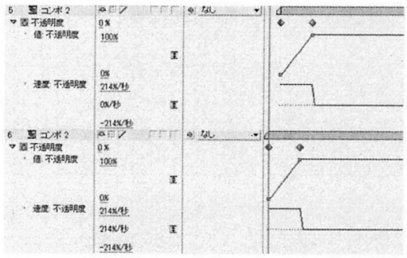

図 4　Adobe After Effects のタイムライン部分

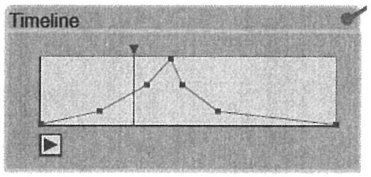

図 5　タイムラインノード

本システムでも同様にキーフレームアニメーションによって複数のパラメータ変化を実現する。After Effects のような既存のシステムは一つのタイムラインしか持っておらず、タイムライン上の再生時間を移動させると、システム全体の再生時間が移動してしまう。これではライブ映像パフォーマンス中に、キーフレームを編集したときにそれらの編集がどのように映像に反映されるかを確認することができない。

本システムでは DF 図の表現を利用し、タイムライン表示部分をタイムラインノードと呼ぶ DF 図上の一つのノードとして複数生成できるように実装した。これらのタイムラインノードそれぞれに再生時間軸を設け、非同期にすることで、あるタイムラインの再生中にでも、そのタイムラインを含む映像経路に影響することなく別のタイムラインを編集し確認することが可能である。タイムラインノードを図 5 に示す。中央部分はキーフレームを表示するタイムラインである。タイムライン上にある縦線は再生している時間を表すインジケータである。左下には再生ボタンを備えている。全てのタイムラインノードはインジケータと再生ボタンを一つづつ持っており、それぞれ別の時間軸上で再生できる。このように複数のタイムラインを非同期にすることで、あるタイムラインの再生中に別のタイムラインを編集することが可能になるが、その編集が終わった後は実際に再生することとなる。その場合には個々のタイ

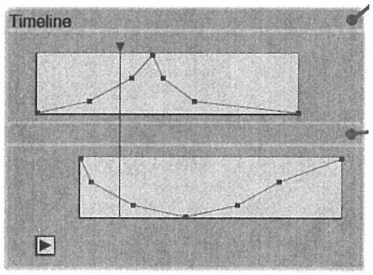

図 6　同期化されたタイムラインノード

ムラインが別々の位置に表示されていると、あるタイムライン上のある時点が別のタイムライン上でどの時点に当たるか把握しづらくなる。そこで、複数のタイムラインノードを必要に応じて同期させることも出来るようにした。あるタイムラインノードを別のタイムラインノードにドラッグアンドドロップすることで二つのタイムラインを備えた一つのノードにまとめることができる。まとめられた二つのタイムラインは一つの再生時間軸を共有しており同時に再生される。また一つのノードにまとめる時に左右にずらした位置にドロップするとそのままの位置でまとめられ、この左右のずれはそのまま時間のずれとして再生タイミングに反映される。図 6 に同期化されたタイムラインノードを示す。二つのタイムラインを持っているが、インジケータと再生ボタンは一つしか持っていない。これは二つのタイムラインが同じ時間軸上で再生されることを表している。

3.2　実装

実装言語は主に C# を使った。映像合成の処理には Microsoft DirectX 9.0 と HLSL (High Level Shader Language) を使用した。扱う映像の形式は 640px × 480px のサイズで 30FPS とした。

映像の各フレームを DirectX のテクスチャとして用意し、それぞれのエフェクトの処理結果は再びテクスチャにレンダリングした。これはエフェクトの入出力を同じ形式で扱うことで処理の内容を DF 図で表現しやすくすることと、処理内容を GPU 上で行うことで高速化できることを期待したためである。

各ノードの映像処理関数の実行は要求駆動で行われる。1/30 秒ごとに映像表示用のノードの描画関数が呼び出され、映像表示用のノードは自分の入力に繋がったノードの映像処理関数を呼び出す。この呼び出しが結線をもった各ノードで再帰的に呼び出され、自分の入力に繋がったノードでの映像処理が終了してから自分の映像処理が実行される。これにより、映像表示用のノードへの経路を持たないノードの映像処理関数は実行しないという、遅延評価を実現している。

DF 図部分の GUI も DirectX によって描画している。これは DF 図上の映像表示用のノードにおいて映像表示を行う時に、グラフィックカード上のテクスチャメモリから処理結果の画像を取得するオーバーヘッドを無くすためである。

3.3　予備実験

まず、1.1 節で述べた作業を本システムで実際に行えるか試した。実装したミキサは複数の映像を重ねて表示するだけであったが、映像の透明度を変更するエフェクタを用意していたので、このエフェクタとミキサを組み合わせることにより、複数の映像経路を切り替えることができた。また、その透明度変更エフェクタのさらに前でエフェクトを適用することで図 1 と同じ構成を構築することも可能であった。この様子を図 2 に示す。さらに、図 2 のスライダノードをタイムラインノードに変更することで、適用したエフェクトのかかり具合を変更しながら映像を切り替えるということも可能であった。

既存システムにはできなかった、より複雑な構成も試した。図 7 では一つの画像を 6 つに分岐させ、それぞれタイミングをずらした回転のアニメーションを適用し、ミキサによってふたたび一つの映像へ合成している（映像 A)。さらにその合成結果を別の映像 B と合成している。映像 A の作成は映像 B を再生したまま行った。これは既存のシステムにはない即興性を示している。

これらの実験には、Intel Core2 Duo 2.66GHz、2GB RAM 及び、NVIDIA GeForce 8800 GTX 768MB のグラフィックカードを備えた計算機を使った。実際のライブ映像パフォーマンスを想定するため、デュアルディスプレイ環境で、メインアウトプットとして一方のディスプレイにフルスクリーンで表示しながら行ったが、アウトプットのフレームレートは常に 30FPS を維持しており、実行速度は十分であった。

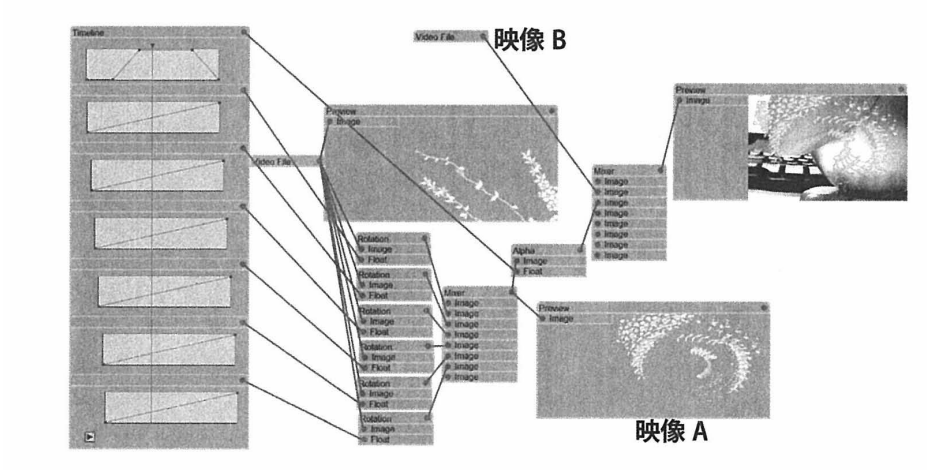

図 7　予備実験で試された複雑な構成

4　まとめと今後の課題

本研究では、DF 図に基づいた映像合成システムの設計、実装を行い、そのシステムによってライブ映像パフォーマンスにおいて必要となる、複数の映像経路の切り替えや、それらのプレビューを行うことが可能であることがわかった。さらに、映像合成システムの構成を再生中にも変更可能にすることで既存のシステムには不可能な、自由でリアルタイム性の高いパフォーマンスの可能性を示した。また、ライブ映像パフォーマンスでタイムラインを扱う機構として、複数の非同期なタイムラインを作成し、それらを同期化する機構を設計、実装した。

以下に今後の課題を述べる。本システムの GUI はまだまだ改善の余地がある。ライブ映像パフォーマンス中には素早く操作することが求められるので、例えば、ノードの追加時に自動的に結線を行うことや、複雑になった DF 図の一部をまとめる機構などが必要である。また、今回実装されたノードの数も少なく、より多くのノード、カメラ入力ノードや実用的なエフェクタノードなどの実装も今後の課題である。そしてある程度実用に耐える実装を行った後、実際にライブ映像パフォーマンスを行うアーティストらの協力によるユーザ評価も行う。

参考文献

1) Bert Bongers and Yolande Harris. A structured instrument design approach: the video-organ. In *NIME '02: Proceedings of the 2002 conference on New interfaces for musical expression*, pp. 1–6, May 2002.
2) Satoru dangkang Tokuhisa, Yukinari Iwata, and Masa Inakage. Rhythmism: a VJ performance system with maracas based devices. In *ACE '07: Proceedings of the international conference on Advances in computer entertainment technology*, pp. 204–207, June 2007.
3) Kentaro Fukuchi, Mertens Sam, and Tannenbaum Ed. Effectv: a real-time software video effect processor for entertainment. In *Entertainment Computing - ICEC 2004*, pp. 602–605, September 2004.
4) Tomoyuki Hansaki, Buntarou Shizuki, Kazuo Misue, and Jiro Tanaka. FindFlow: visual interface for information search based on intermediate results. In *APVis '06: Proceedings of the 2006 Asia-Pacific Symposium on Information Visualisation*, pp. 147–152, February 2006.
5) Wesley M. Johnston, J. R. Paul Hanna, and Richard J. Millar. Advances in dataflow programming languages. *ACM Computing Surveys*, Vol. 36, No. 1, pp. 1–34, March 2004.
6) Michael Lew. Live cinema: designing an instrument for cinema editing as a live performance. In *NIME '04: Proceedings of the 2004 conference on New interfaces for musical expression*, pp. 144–149, June 2004.
7) Satoru Tokuhisa. Nervixxx. In *Laval Virtual Award 2008*, April 2008.
8) 小林敦友, 志築文太郎, 田中二郎. リアルタイム映像パフォーマンス向け映像合成システム. 情報処理学会第 70 回全国大会講演論文集, 第 4 巻, pp. 157–158, March 2008.