J-012

動的に更新される背景バッファを用いた高画質多視点 VoD システムのソフトウェア実装

Software Implemenatation of Free Viewpoint Image Synthesizing System Using Dinamically-Updated Background Buffers

松村 篤志 † 川田 亮一 † 小池 淳 † 松本 修一 †
OB Atsushi Matsumura Ryoichi Kawada Atsushi Koike Shuichi Matsumoto

1. まえがき

次世代映像アプリケーションの 1 つとして，自由視点映像に注目が集まっている．自由視点映像とは，1 つの映像素材をもとに，ユーザが任意に選択した視点からの映像を描画するシステムである．ユーザの視点は無数に存在するため，これら全てに対応した素材をあらかじめ用意しておくことは現実的ではない．そこで，3 次元情報が記述された素材を用いて他視点からの映像を推定する手法が一般的である．

代表的な手法の 1 つに，一視点からの映像 (以下参照画像と表記) と奥行き情報 (以下奥行きマップ) を用いる手法がある [1]．同手法には，奥行きマップのデータ量が参照画像と比べて小さく，また従来の符号化方式との親和性が高いなどの長所がある．しかしながら，隠蔽された背景に関する情報が存在しないため，視点を移動した際に描画不能となる領域が生じてしまう．この問題を解決するため，先に筆者らは背景バッファをフレーム毎に更新し，これを用いて描画不能な画素に対して補間を行う手法を考案している [2]．

同手法を適用したアプリケーションの普及には，専用ハードウェアなどを必要とせず，安価なソフトウェアで実装可能であることが望ましい．そこで，先の考案手法に改良を加え，ソフトウェア実時間処理が可能な多視点 VoD システムを開発した．本稿では，これについて述べる．

2. 自由視点映像の生成手順

文献 [2] における各フレームの処理手順を図 1 に示す．まず，参照画像と奥行きマップのみを用いてユーザが所望する視点からの画像を生成し，これを仮自由視点画像と定義する．その一方で参照画像から背景領域のみを抽出し，これを用いて背景バッファの更新を行う．最後に，同バッファを用いることによって，仮自由視点画像内の描画不能画素に対する補間を行い，結果をユーザに出力する．以下の節において各手順についての説明を行う．

2.1 仮自由視点画像の生成

自由視点画像を得るためには，ユーザの視点を定めるパラメータが必要となる．同パラメータを，回転量 ψ, θ, ϕ, 平行移動量 t_x, t_y, t_z として定義する．これらのパラメータを用いることによって，参照画像 I と仮自由視点画像 A における画素間の対応点 (u, v), (u'', v'') が以下の式により求まる．

$$(u'', v'', 1) \times \left(D_I(u, v)\mathbf{R}_\phi\mathbf{R}_\theta\mathbf{R}_\psi(u, v, 1)^T + t\right) = 0 \quad (1)$$

† (株) KDDI 研究所, KDDI R&D Inc.

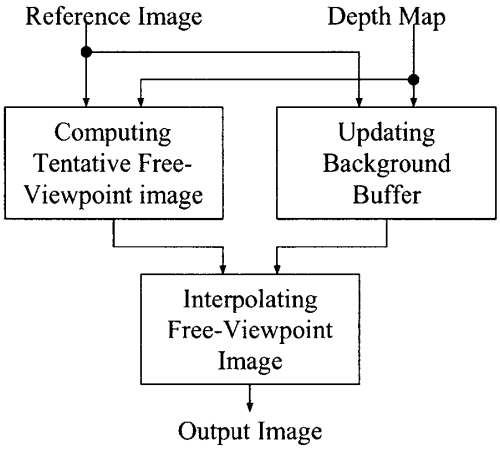

図 1: 自由視点映像補完の流れ

ただし，

$$\mathbf{R}_\phi = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2)$$

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \quad (3)$$

$$\mathbf{R}_\psi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \quad (4)$$

$$t = [t_x, t_y, t_z]^T \quad (5)$$

であり，$D_I(u, v)$ は，画像 I 内の座標 (u, v) における奥行きを表す．

2.2 背景バッファの生成および更新

参照画像と奥行きマップを用いて背景バッファ U を生成，更新する手順を以下に示す．前処理として，参照画像における奥行きが閾値以上となる画素のみを抽出し，同画素のみを用いた背景画像 J を得る．先頭フレームにおいては，J を U にそのまま投影する．その他のフレームにおいては，J, U 内における対応点のサンプル (u_J, v_J), (u'_U, v'_U) を 8 点以上求め，これらを用いて以下の式を満足する行列 $\mathbf{B}$ を算出する．

$$(u_J, v_J, 1)^T \times \mathbf{B}(u'_U, v'_U, 1)^T = 0 \quad (6)$$

その後，J および U 内の全画素に対して式 (6) を適用することによって対応点を算出した上で，以下の式によっ

て背景バッファの更新を行う．ただし，$U(u_J, v_J)$ は画像 U 内における座標 (u_J, v_J) の画素値を表し，以下各画像の画素値を同様の表記によって表す．

$$U(u_J, v_J) \leftarrow \begin{cases} U(u'_U, v'_U) & \text{if } J(u_J, v_J) = \text{null} \\ J(u_J, v_J) & \text{otherwise} \end{cases} \quad (7)$$

ここで，左矢印は，右辺の値を左辺に代入することを表す．同式は，図 4 にて示すように，各フレームにて生成される背景画像を基準とし，それに対して前フレームにて生成された画像を合成することを表す．

2.3 自由視点画像の補完

2.2 節にて生成された背景バッファを用いて，以下の手順によって仮自由視点画像の補完を行う．まず，A, U 内における対応点のサンプル (u''_A, v''_A), (u'_U, v'_U) を 8 点以上求め，以下の式を満足する行列 $\mathbf{B}'$ を算出する．

$$(u''_A, v''_A, 1)^T \times \mathbf{B}'(u'_U, v'_U, 1)^T = 0 \quad (8)$$

その後，A および U 内の全画素に式 (8) を適用することで対応点を算出した後，以下の式により出力画像 O を得る．

$$O(u''_A, v''_A) = \begin{cases} U(u'_U, v'_U) & \text{if } A(u''_A, v''_A) = \text{null} \\ A(u''_A, v''_A) & \text{otherwise} \end{cases} \quad (9)$$

3. ソフトウェア実装のため検討

本章では，**2.** 章の手順をソフトウェア上にて実装する際に行った考察，および改善について述べ，最後に同改善を適用したアプリケーションの概略について述べる．

3.1 実時間処理への適応

実装に際し，**2.2** 節における $\mathbf{B}$ を算出するための計算コストが大きく，実時間処理が困難となる．そこで，$\mathbf{B}$ の算出は視点位置に依存しないことに着目し，同問題に対する解決手段を図 2 に示す．まず，各フレームにおける $\mathbf{B}$ をあらかじめ算出しておき，その構成要素をファイルに蓄積しておく．画像を生成する際には，参照画像，奥行きマップとともにこれらの要素を既知の値として入力し，これをもとに $\mathbf{B}$ の再構築を行う．$\mathbf{B}$ は 9 つの実数値によって構成されているため，そのデータ量は参照画像や奥行きマップに比べ極めて小さい．従って，素材全体のサイズを大幅に変更させることなく，同改善を取り入れることが可能となる．

改善効果を検証するため，各フレームの描画時間を計測した．結果を図 3 に示す．x 軸にフレーム番号を，y 軸に各フレームの描画時間を表す．ただし，**2.2** 節における対応点の探索には勾配法 [3] を用いた．$\mathbf{B}$ の算出を描画時に行った場合には，各フレームの描画時間は平均 2.178 秒であったが，改善手法を用いることによって，平均 0.053 秒となった．すなわち，描画速度が約 41 倍に向上されたこととなる．

3.2 視点の移動と回転の連動

式 (1) に必要なパラメータは ϕ, θ, ψ, および t_x, t_y, t_z の 6 つであり，これらの設定を全てユーザに依存した場合，操作が煩雑になってしまう．そこで本ソフトウェア

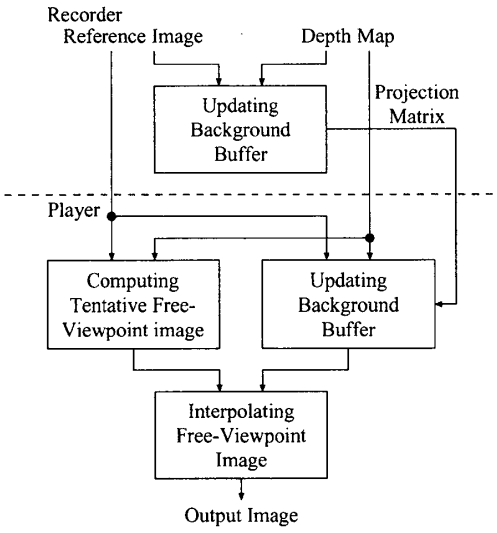

図 2: 実時間処理のための流れ

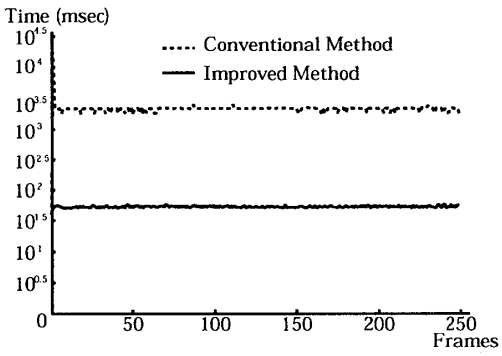

図 3: 各フレームにおける描画時間

では，以下の式を用いることにより，ϕ, θ, ψ と t_x, t_y との関連づけを行う．

$$t_x = M_d(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi) \quad (10)$$

$$t_y = M_d(\sin\phi \sin\theta \cos\psi + \cos\phi \sin\psi) \quad (11)$$

ただし，M_d は奥行きの最大値と最小値の平均を表す．同式を用いることによって参照画像と自由視点映像の画素の中心が一致するため，常に参照画像の存在する方向に視点を移動させることが可能となる．さらに，ϕ 方向における回転が視覚に与える影響は少ないと考え，$\phi = 0$ に固定させた．これらを実装することで，ユーザに依存するパラメータは θ, ψ, および t_z の 3 つのみとなり，操作の簡易化をはかることができる．

3.3 実装ソフトウェア

開発に使用した計算機環境を表 1 に，実行画面を図 5 に示す．

同ソフトウェアでは，参照画像と奥行きマップをファイルから指定し，これらを用いて自由視点映像の生成を行う．同ファイルとして，4:2:2 の YUV 形式，および MPEG4 形式の画像を用いることができる．また，パラ

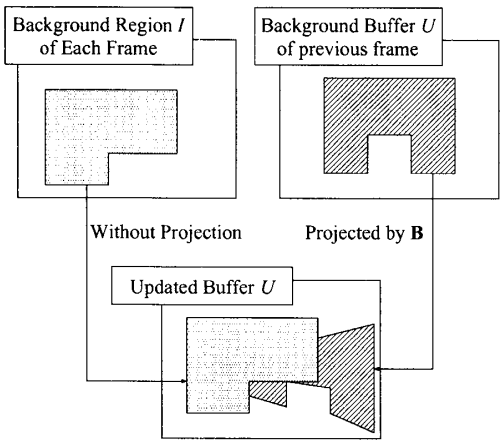

図 4: 背景バッファの更新

表 1: ソフトウェア開発環境

項目	スペック
CPU	Pentium4 2.4GHz
メモリ	1GBytes
OS	Windows XP Professional
開発言語	VisualC++ 6.0

メータの設定は，θ, ψ はマウスを用いた画像領域内のドラッグにて，同様に t_z スクロールバーのドラッグにて行う．生成された映像はウィンドウ左下部に描画される．ただし，図 5，および 6 では参照画像の範囲外ないし隠蔽に起因し，描画不能となった画素は黒色にて出力している．

出力の一例として，MPEG4 3DAV 標準画像の 1 つである Interview を素材とし，補完がなされた場合となされなかった場合の第 250 フレームをそれぞれ図 6(a)，(b) に示す．同結果より，補完を用いることによってによって，描画不能画素が減少していることがわかる．

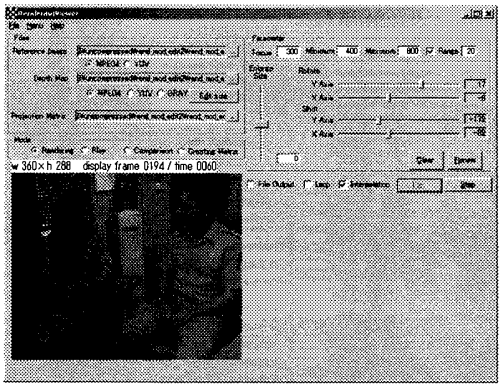

図 5: ソフトウェア実行画面

(a) 補完あり

(b) 補完なし

図 6: Interview における第 250 フレーム

4. まとめ

本稿では，高画質な多視点 VoD システムのソフトウェア実装について述べた．同ソフトウェアでは，出力画像を描画する際に，動的に更新される背景バッファを用いた補完を行うことで高画質化を実現している．実装上の工夫により，約 41 倍の高速化を達成し，その結果ソフトウェアによる実時間描画が可能となった．また，視点位置の回転と移動を連携させることにより，平易な操作による視点移動を実現させた．今度の課題として，PDA に代表される携帯端末への実装を実現することが挙げられる．

参考文献

[1] C. Fehn, K. Schuur, P. Kauff and A. Smolic, "Meta-Data Requirements for EE4 in MPEG 3DAV", ISO/IEC JTC1/SC29/WG11, MPEG02/M9559, Pattaya, Thailand (Mar. 2003)

[2] 松村 篤志，内藤 整，川田 亮一，小池 淳，松本 修一，"動的に更新される多層背景バッファを用いた高精度な自由視点映像生成法式" 映像情報メディア学会誌 Vol. 58, No. 6, Jun. 2004. (採録決定)

[3] 川田 亮一，浜田 高宏，松本 修一，"動き補正テレビ方式変換の改善"，映像情報メディア学会誌，Vol. 51, No. 9, pp.1577–1586, Sep. 1997.