Vol.2011-CG-142 No.2
2011/2/8

時系列ボリュームレンダリングの描画速度を調節するための非可逆圧縮手法

山田真義[†1]　伊野文彦[†1]　萩原兼一[†1]

本稿では，時系列ボリュームの描画速度を調節することを目的として，非可逆圧縮手法を提案する．提案手法は，データの読み出しが描画性能を決めることに着目し，データ圧縮率を制御することにより描画速度を調節する．この実現のために，既存の非可逆圧縮手法 PVTC （Packed Volume Texture Compression）を拡張し，時系列ボリュームの持つ空間類似性および時間類似性の観点から圧縮率を選択可能にする．この選択機能は，NVIDIA 製 GPU 向けの統合開発環境 CUDA を用いて実装されている．提案手法を 2 種類のデータセットに適用した結果，提案手法は PVTC と比較して，画質を維持したまま描画速度を 1.45 倍ほど向上できた．逆に，描画速度を 25%ほど低下させることにより，PSNR 値を 10〜20%ほど向上できた．

Lossy Compression Method for Controlling Rendering Speed of Time-Varying Volume

Masayoshi Yamada,[†1] Fumihiko Ino[†1]
and Kenichi Hagihara[†1]

In this paper, we present a lossy compression method aiming at controlling rendering speed of time-varying volume. Our method controls the rendering speed by varying data compression ratio because the data transfer time determines the rendering performance. To achieve this, we extend an existing lossy compression method called the packed volume texture compression (PVTC) such that it can select the compression ratio in terms of temporal and spatial coherences inherent in a time-varying volume. Such a selection capability is implemented using compute unified device architecture (CUDA), which is a development framework for the NVIDIA GPU. Evaluation results obtained using two datasets show that the rendering speed achieved by our method is 1.45 times higher than that of PVTC without decreasing image quality. Conversely, our method increases the peak signal-to-noise ratio (PSNR) value by 10–20% by decreasing the rendering speed by 25%.

1. はじめに

時系列 VR（Volume Rendering）とは，時系列の 3 次元データ（時系列ボリューム）を順に可視化する技術である．この技術は，物理分野や医用分野などにおいて用いられている．例えば，数値シミュレーションが時間発展する様子や X 線 CT（Computed Tomography）装置により取得した臓器の動きを動画として表示できる．

時系列でない（1 時刻分の）ボリュームに対する VR は，現在の描画用ハードウェア GPU（Graphics Processing Unit）[1] を用いて容易に高速化できる．一方，時系列ボリュームは時間および空間からなる 4 次元の情報であり，そのデータサイズは大きい．例えば，ボクセルあたり 4 バイトの情報を持ち，512^3 ボクセルからなるボリュームが 100 時刻ほどある場合，そのデータサイズは 50 GB に達する．したがって，時系列ボリュームの全体をビデオメモリに格納することは容易ではない．

そこで，既存手法[2,3] はディスクなどの 2 次記憶装置に時系列ボリュームを格納し，現在の描画を終えるたびに次時刻のボリュームをビデオメモリへ転送している．この場合，描画のたびにディスクからのデータの読み出しが必要であり，ディスクからビデオメモリへの転送処理が描画処理の性能ボトルネックとなる．この性能ボトルネックを除去するために，既存手法はボリュームのデータサイズを削減する圧縮手法を描画処理に組み込んでいる．

Nagayasu ら[2] は，時系列ボリュームが空間類似性および時間類似性を持つことに着目し，非可逆圧縮手法 PVTC（Packed Volume Texture Compression）を提案している．PVTC は，GPU がハードウェアとして備える圧縮機能を応用していて，その圧縮率は 1/6 である．具体的には，ボリュームを複数のサブボリュームに分割し，サブボリュームごとに空間類似性に着目した圧縮を施す．この際，3 時刻分のサブボリュームをまとめて圧縮することにより，時間類似性も活用できる．Du ら[3] は木構造に基づく圧縮手法を提案している．この手法は，データ展開時の局所性を高めるために，空間類似性および時間類似性の圧縮順序を決める．

一方，再現性ではなく描画速度を調節したいという要求もある．これは，数値シミュレーションでは，描画速度が大きすぎてもボリュームの詳細を理解できず，小さすぎても時系列

†1 大阪大学大学院情報科学研究科コンピュータサイエンス専攻
Department of Computer Science, Graduate School of Information Science and Technology, Osaka University

ボリュームの全体像を把握することが難しいためである．また時系列 VR では，描画速度が大きい場合は画質が低くても細部の違いは視認できず，逆に小さい場合は高い画質が必要となる．そのために，速度に応じて画質を変える必要がある．しかし，ハードウェアの固定機能による従来の圧縮は圧縮率が固定であった．

本稿では，時系列ボリュームの描画速度を調節することを目的として，非可逆圧縮手法を提案する．提案手法は PVTC を拡張し，ボリュームの圧縮率を可変とする．その実現のために，NVIDIA 社の統合開発環境 CUDA（Compute Unified Device Architecture）[4] を用い，圧縮対象となるサブボリュームの大きさおよび時刻分をデータ圧縮時に指定できる拡張を施す．この拡張により，ボリュームの圧縮率を介して，描画速度を調節できる．例えば，低い圧縮率を指定すれば，2 次記憶装置からビデオメモリへのデータ転送量が減少し，描画を高速化できる．逆に，高い圧縮率を指定すれば，データ損失が減少し，再現性が高まる．これらの特性をうまく使用すれば，再現性を低下させることなく描画を高速化したり，描画速度を低下させて再現性を向上できる．

以降では，2 章で既存手法 PVTC の概要を示す．その後，3 章で提案手法を説明し，4 章で評価実験の結果を示す．最後に，5 章で今後の課題とともに本稿をまとめる．

2. 圧縮手法 PVTC（Packed Volume Texture Compression）

PVTC[2] は，ボリュームに対する圧縮機構 VTC（Volume Texture Compression）[5] を基に，時系列ボリュームへの拡張を施した非可逆圧縮手法である．VTC は GPU のハードウェア機能として実装されているため，データ展開が迅速である．PVTC を用いる VR では，あらかじめ前処理として時系列ボリュームを圧縮する必要がある．以降，圧縮後のボリュームのことを圧縮ボリュームと呼ぶ．描画時には，その圧縮ボリュームを 2 次記憶装置からビデオメモリに転送し GPU で展開する．前述の通り，ハードウェアが展開を担当するため，展開処理のためにプログラムを記述する必要はない．描画のためのカーネルは，単に圧縮テクスチャとしてテクセルを参照すればよい．

図 1 に PVTC の概略図を示す．この圧縮手法は，各ボクセルが 1 バイトのスカラ値を持つことを前提とする．ボリュームを $4 \times 4 \times 1$ ボクセルのサブボリュームに分割し，3 時刻分のサブボリュームをまとめて圧縮する．以降では，サブボリュームの高さ（4×4 ボクセル領域の厚み）を S とし，圧縮対象としてまとめる時刻の数を T と表記する．PVTC では $\langle S, T \rangle = \langle 1, 3 \rangle$ である．圧縮ボリュームはサブボリュームごとの圧縮結果を持つ．各々の圧縮結果は，代表色およびルックアップテーブルの部分からなる．代表色は，時刻 t〜$t+2$ に

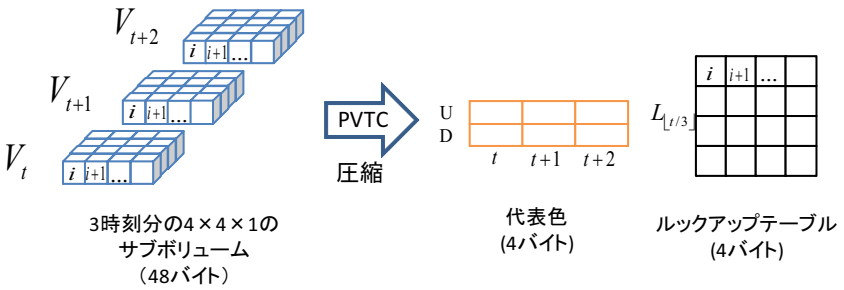


図 1 圧縮手法 PVTC の概略図

おけるサブボリュームの最大値 U_t〜U_{t+2} および最小値 D_t〜D_{t+2} を合わせて 4 バイトで表す．一方，ルックアップテーブルは U_t〜U_{t+2} および D_t〜D_{t+2} から元のボクセル値を近似するための情報を持つ．ルックアップテーブルは，2 ビットの情報を 4×4 個ほど保持していて，その大きさは 4 バイトである．PVTC は，時刻 t（> 0）におけるサブボリューム内の i（$0 \leq i \leq 15$）番目のボクセル値 $V_{i,t}$ を式 (1) に基づいて近似する．

$$V_{i,t} = \begin{cases} U_t, & \text{if } L_{i,\lfloor t/3 \rfloor} = 0, \\ 2U_t/3 + D_t/3, & \text{if } L_{i,\lfloor t/3 \rfloor} = 1, \\ U_t/3 + 2D_t/3, & \text{if } L_{i,\lfloor t/3 \rfloor} = 2, \\ D_t, & \text{if } L_{i,\lfloor t/3 \rfloor} = 3. \end{cases} \tag{1}$$

ここで，$L_{i,\lfloor t/3 \rfloor}$ はルックアップテーブルが持つ 2 ビットの情報に対応する．

式 (1) は，PVTC の展開処理がサブボリュームごとに独立であることを示している．したがって，並列処理に向く．また，各ボクセルは同一の演算に基づいて展開できるため，SIMD（Single Instruction Multiple Data）型のアーキテクチャに適している．これらの特性は GPU による高速化に貢献する．

3. 描画速度を調節するための圧縮手法

時系列 VR は 2 次記憶から主記憶への転送時間が描画速度の性能ボトルネックであるので，転送時間を短縮することにより描画速度を高速化できる．そこで我々はボリュームの圧縮率を可変にし制御することで，圧縮ボリュームの転送量を調節することにより，描画速度の調節を目指す．また，転送時間はボリュームサイズに比例するので，圧縮前に転送量が定

まることにより推測できる．

したがって，描画速度の調節を実現する圧縮方式は以下の 3 点を満たす必要がある．

条件 1：圧縮率が可変であること

条件 2：あらかじめ圧縮後のファイルサイズが定まること

条件 3：展開に要する時間が短いこと

条件 1 が必要である理由は，圧縮ボリュームの転送量を調節し，描画速度を調節するためである．次に，条件 2 が必要である理由は，転送時間を事前に予測できる必要があるためである．圧縮後のファイルサイズが定まらない場合，転送時間を推測できずに描画時間を調節することができない．最後に，条件 3 が必要である理由は，圧縮ボリュームの展開が描画速度の性能ボトルネックにならないためである．圧縮ボリュームの展開が性能ボトルネックになれば，転送量を制御することにより描画速度を調節できない．

これらの条件を満たすために，永安らの提案する圧縮手法 PVTC を用いる．PVTC は非可逆圧縮であり，圧縮率固定である．このため，圧縮後のボリュームサイズが定まり，条件 2 を満たす．また PVTC は，GPU のような SIMD 型のアーキテクチャに適し，計算量が少ないため，条件 3 を満たす．しかし，PVTC は GPU のハードウェアの固定機能を用いる圧縮であるために条件 1 を満たさない．そこで我々は，統合開発環境 CUDA を用いて PVTC の圧縮率を可変にし制御することで，描画速度を調節する圧縮手法を提案する．

3.1 圧縮率が可変である PVTC

我々は，PVTC のサブボリュームの形状および圧縮する時刻分を変化させることにより圧縮率を可変にすることを提案する．そのために時系列ボリュームの特性である空間類似性と時間類似性に着目し，PVTC のサブボリュームの形状と圧縮する時刻分を可変にする．可変にするために，PVTC と比較してより空間的類似性を重視し，空間方向の圧縮を高めて圧縮率を低くする 4×4×4 の形状（$S = 4$）および時間方向の圧縮を緩めて，圧縮率を高める時刻分が 2（$T = 2$）の圧縮方法を用意する．$S = 4$ では，ルックアップテーブルを $S = 1$ と同様に各 1 ボクセルを 2 ビットで表し，合計 16 バイトで情報を保持する．このとき，64 ボクセルを 1 つの代表色で近似するために，$S = 1$ と比較すると，再現性は低下する．$T = 2$ では，代表色を $T = 2$ と同様に 4 バイトで表現し，1 時刻の代表色の情報を H_t および L_t は 8 ビットで保持する．ゆえに，$T = 3$ と比べて最大値が $2^5 = 32$ から $2^8 = 256$ まで増加し，再現性が増加する．この S および T を組み合わせて 4 種類用意することにより圧縮率が可変となり（条件 1），パラメータを決定することで**表 1** に示すように圧縮率は一意に定まる（条件 2）．圧縮率が一意に定まることにより圧縮ボリュームのファイルサイズが制御できるので，性能ボトルネックとなる転送時間を制御し，描画時間を調節できる．

表 1 圧縮パラメータおよび圧縮率の関係

パラメータ $\langle S, T \rangle$	圧縮率（%）	サブボリュームの形状	圧縮時刻分
$\langle 4, 3 \rangle$	10.4 （5/48）	4×4×4	3
$\langle 4, 2 \rangle$	15.6 （5/32）	4×4×4	2
$\langle 1, 3 \rangle$	16.7 （1/6）	4×4×1	3
$\langle 1, 2 \rangle$	25.0 （1/4）	4×4×1	2

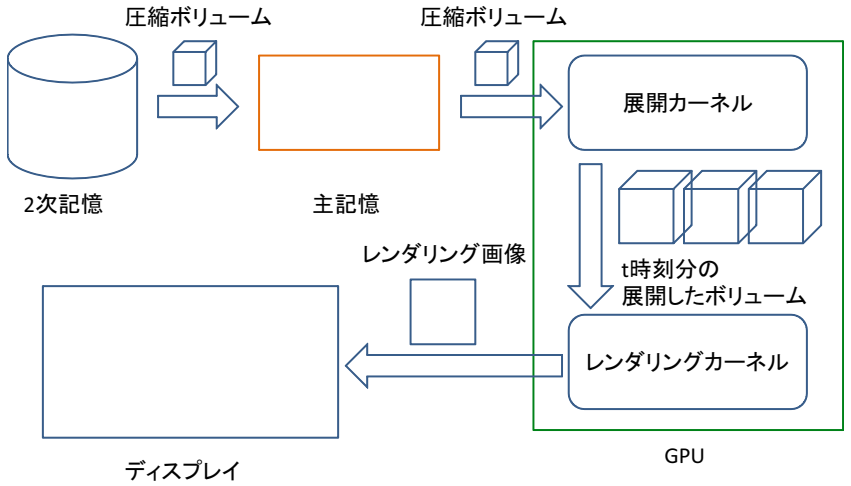


図 2 描画の一連の流れ

また，PVTC は式 (1) に示したようにサブボリュームごとに計算が独立であり，サブボリューム数が多いので，GPU などの SIMD 型の並列計算機に適した圧縮方式である．さらに，展開に生じる計算のアルゴリズムは単純で計算量が少ないために，展開時間は短い（条件 3）．

3.2 実　　装

描画の一連の流れを**図 2** に示す．あらかじめ前処理により圧縮した t 時刻分の圧縮ボリュームを 2 次記憶から主記憶に転送し，GPU に転送する．そして，GPU で展開カーネルを呼び出し圧縮ボリュームを展開し，t 時刻分の展開ボリュームを生成する．そこで生成された展開ボリュームをレンダリングカーネルが読み込みレンダリングし，ディスプレイに描画す

る．このレンダリングカーネルは，CUDA SDK の volumeRender を用いる[4]．条件 3 を満たすためには，展開カーネルが短時間で終了しなければならない．

そのために我々は，圧縮ボリュームのデータ構造およびスレッドの割り当てに着目する．CUDA では，16 スレッドを 1 つのまとまりであるワープとして同時に命令を実行し，メモリアクセスもワープ単位で実行する．このメモリアクセスは非連続アドレスを参照する場合と比較して，連続アドレスを参照する場合の方が高速である．その高速化ために，圧縮ボリュームは，展開時に連続アドレスを参照するデータ構造が必要であり，同時に，展開カーネルが連続アドレスを参照する必要がある．

圧縮ボリュームは，サブボリュームごとの代表値およびルックアップテーブルからなる．展開時に 1 ワープが連続領域を参照する必要があるために，圧縮ボリュームの代表値およびルックアップテーブルを完全に分離し，各々が連続であるデータ構造にする．パラメータ $\langle S, T\rangle$ であれば，各サブボリュームの代表値が 4 バイトおよびルックアップテーブルが $4S$ バイトで構成される．したがって，圧縮ボリュームの前半 $4/(4+4S) = 1/(1+S)$ が代表値の保存領域であり，後半 $4S/(4+4S) = S/(1+S)$ がルックアップテーブルの保存領域とする．このように実装することにより，圧縮ボリュームは展開時に連続アドレスを参照するデータ構造となる．

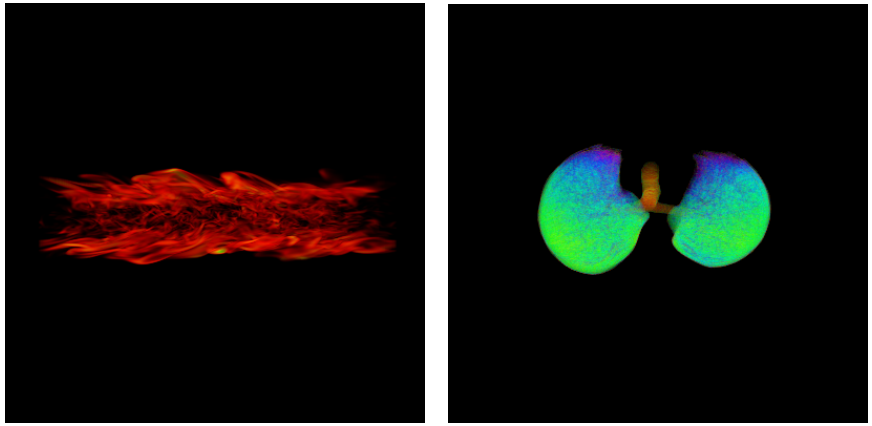
スレッドは，展開カーネルが連続アドレスを参照するように割り当てる．そのためにスレッドは 1 つのサブボリュームを展開する．これにより，各スレッドが代表色 4 バイトおよびルックアップテーブル 16 バイトを参照し，各ワープが 64 バイトおよび 256 バイトの連続領域を参照する．このメモリアクセスは CUDA を用いる場合，効率のよいメモリアクセスが出来る．もし 1 つのスレッドがサブボリュームより小さい領域を担当する場合，サブボリューム内で共通の計算が生じるために，無駄な計算が生じる．逆に，1 つのスレッドが複数のサブボリュームを担当する場合，スレッド数が減少するために並列度が落ちる可能性がある．

4. 評価実験

提案手法を評価するために，描画速度および画質に関して既存手法 PVTC と比較した結果を示す．実験に用いた計算機は CPU として Intel Xeon X5450（3.0 GH z）を持ち，GPU として NVIDIA GeForce GTX 580 を持つ．また，主記憶およびビデオメモリの容量はそれぞれ 8 GB および 1.5 GB であり，2 次記憶装置として OCZ Colossus Series SSD 500 GB を搭載する．計算機の OS は 64 ビット版の Windows XP であり，ビデオドライバのバージョンは 262.99 である．実験に用いた CUDA のバージョンは 3.0 である．

表 2 実験で用いたデータセット

データセット	ボリュームサイズ（ボクセル）	データサイズ（MB/時刻）	時刻数
乱流	$480 \times 720 \times 120$	39.6	60
肺	$512 \times 512 \times 296$	74.0	72



(a) 乱流　　(b) 肺

図 3 レンダリング結果（投影像サイズ：512×512 ピクセル）

時系列ボリュームとしては，ジェット気流のシミュレーション結果[6] および X 線 CT 装置により取得した肺に対する位置合わせの過程[7] を用いた．前者は空間類似性および時間類似性ともに低く，後者は空間類似性および時間類似性ともに高い．**表 2** に，実験に用いたデータセットの概要を示す．表に示したデータサイズは 1 時刻あたりの値である．すべての時刻を含めた全体のデータサイズは，乱流および肺の各々について 2.3 GB および 5.2 GB である．今回の実験機では，いずれもビデオメモリ上に全体を格納することは不可能である．各々のレンダリング結果を**図 3** に示す．投影像のサイズは 512×512 ピクセルである．

4.1 描画速度評価

パラメータ $\langle S, T\rangle$ の指定により描画速度を調節できているか否かを確認するために，1 秒あたりに描画できたボリュームの数 F を計測した．実験では，1 回の描画を終えるたびに時刻を 1 つ進め，描画を繰り返した．**表 3** に各データセットの描画速度 F を示す．なお，表

表 3 各データセットの描画速度 F（ボリューム数/秒）

データセット	パラメータ $\langle S,T\rangle$			
	$\langle 4,3\rangle$	$\langle 4,2\rangle$	$\langle 1,3\rangle$	$\langle 1,2\rangle$
乱流	25.5	19.4	17.6	13.7
肺	14.8	11.2	10.1	7.6

表 4 描画時間の内訳（ミリ秒）

内訳	乱流				肺			
	パラメータ $\langle S,T\rangle$				パラメータ $\langle S,T\rangle$			
	$\langle 4,3\rangle$	$\langle 4,2\rangle$	$\langle 1,3\rangle$	$\langle 1,2\rangle$	$\langle 4,3\rangle$	$\langle 4,2\rangle$	$\langle 1,3\rangle$	$\langle 1,2\rangle$
b_1	25.1	36.4	41.7	57.4	48.8	68.7	79.3	105.4
b_2	2.2	3.2	3.4	5.1	3.9	5.9	6.2	9.3
b_3	0.6	0.6	0.7	0.9	1.0	1.1	1.2	1.5
b_4	6.2	6.2	6.2	6.2	10.8	10.8	10.8	10.8
b_5	6.0	5.7	6.4	5.2	4.7	5.3	4.9	5.5
描画時間 $1/F$	39.2	51.5	56.8	73.2	67.7	89.0	98.4	131.0

中の値は全時刻分の平均値である．圧縮率が高いパラメータほど描画速度 F が大きい．したがって，圧縮率を基に描画速度 F を調節できる．

次に，圧縮ボリュームの展開に要するオーバヘッドを評価するために，時刻分 1 つあたりの描画時間 $1/F$ の内訳 b_1〜b_5 を調べた（**表 4**）．ここで，b_1 は 2 次記憶装置から主記憶までの転送時間であり，b_2 は主記憶からビデオメモリまでの転送時間である．また，b_3 および b_5 は展開カーネルおよび描画カーネルの実行時間を表し，b_4 はバインドなどのテクスチャ操作に要する時間を表す．なお，内訳の計測にはオーバヘッドが生じるため，$b_1+b_2+\cdots+b_5 > 1/F$ であることに注意されたい．

圧縮ボリュームの展開に要する時間 b_3 は，描画時間 $1/F$ の 1〜2%に過ぎず，展開オーバヘッドは小さい．したがって，提案手法は 3 章で示した条件 3 を満たしている．また，転送時間 b_1 および b_2 は圧縮率に比例している．これらの転送時間は描画時間 $1/F$ の 69〜88 %を占めていて，描画速度 F を決めている．ゆえに，圧縮ボリュームの圧縮率を指定することにより描画速度を制御できている．

4.2 画質評価

提案手法により得られるレンダリング結果の画質を評価するために，圧縮を施さずに得られる結果との比較を示す．画質の指標として，PSNR（Peak Signal-to-Noise Ratio）を用いた．PSNR は非可逆圧縮の再現性を評価する指標として広く使用されている．

表 5 各データセットの PSNR 値

データセット	パラメータ $\langle S,T\rangle$			
	$\langle 4,3\rangle$	$\langle 4,2\rangle$	$\langle 1,3\rangle$	$\langle 1,2\rangle$
乱流	31.5	39.2	31.8	40.9
肺	33.9	37.4	36.7	40.8

表 5 に各データセットの PSNR 値を示す．表中の値は全時刻分の平均値である．既存手法 PVTC が用いるパラメータ $\langle S,T\rangle = \langle 1,3\rangle$ を $\langle 4,3\rangle$ および $\langle 1,2\rangle$ と比較すると，いずれも圧縮率が低いほど PSNR 値は小さい．これは PVTC と比較して，$\langle 4,3\rangle$ は圧縮する単位であるサブボリュームを大きくしたために PSNR 値は減少し，$\langle 1,2\rangle$ はサブボリュームの圧縮時間分を短くしたために PSNR 値が上がる．次に $\langle S,T\rangle = \langle 1,3\rangle$ および $\langle 4,2\rangle$ を比較する．これらを比較すると，圧縮率はパラメータ $\langle 4,2\rangle$ が低いが，PSNR 値は大きい．これはパラメータ $\langle 4,2\rangle$ は，$\langle 1,3\rangle$ と比較して，サブボリュームは大きく，圧縮時間は短い．そのために，$\langle 1,3\rangle$ と $\langle 4,2\rangle$ の再現性は，圧縮率では単純比較できずにボリュームデータの特性によって値が異なる．肺のボリュームデータでは時間類似性が高いために $\langle 1,3\rangle$ および $\langle 4,2\rangle$ は PSNR 値にあまり変化がなく，乱流のボリュームデータでは時間類似性が低いために，大きく変化したと考えられる．

最後に，目視による定性的な評価を示す．**図 4** は，乱流データの 30 時刻目の非圧縮ボリュームと各パラメータで圧縮したボリュームの拡大したレンダリング結果である．この時刻を選んだ理由は，乱流データが 60 時刻のボリュームデータであるので，その中央の時刻であり，一番動きが大きい時刻であるためである．

この図を見ると，パラメータ $\langle 4,3\rangle$ と $\langle 1,3\rangle$ の画像は，$\langle 4,2\rangle$ や $\langle 1,2\rangle$ と比べ，中央が非圧縮のものと比べて暗くなっており詳細が見にくいことが分かる．$\langle 4,2\rangle$ や $\langle 1,2\rangle$ は，非圧縮のものと比べても大差のないレンダリング結果となっており，$\langle 4,2\rangle$ は $\langle 1,3\rangle$ と比べて圧縮率は低く描画速度は大きいが，良いレンダリング結果を示した．これらは PSNR 値と同様の傾向を示している．

5. まとめ

本稿では，時系列ボリュームの描画速度を調節することを目的として，時系列ボリュームに対する非可逆圧縮手法を提案した．提案手法は統合開発環境 CUDA を用い，既存の圧縮手法 PVTC を拡張する．具体的には，圧縮対象となるサブボリュームの大きさおよび時刻分をデータ圧縮時に指定することにより，圧縮率を介して描画速度の調節を図る．また，

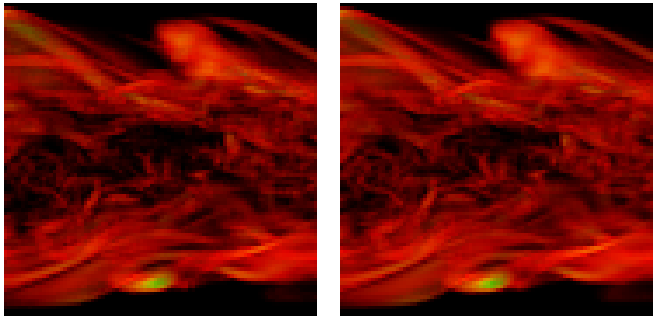
展開のためのオーバヘッドを小さくなるように，圧縮ボリュームのデータ構造を決め，各スレッドに計算を割り当てる．その結果，PVTC と比較して，画質を維持したまま描画速度を 1.45 倍ほど向上できた．逆に，描画速度を 25%ほど低下させることにより，PSNR 値を 10〜20%ほど向上できた．

今後の課題としては，空間方向や時間方向のパラメータを増やし圧縮率の自由度を高めることが挙げられる．

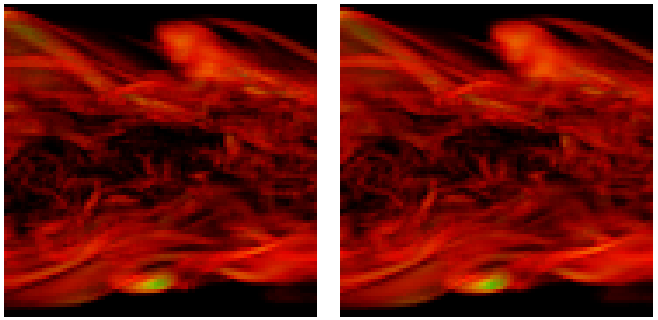
謝辞 本研究の一部は，科学研究費補助金基盤研究（A）（20240002）および大阪大学グローバル COE プログラム「予測医学基盤」の補助による．

参 考 文 献

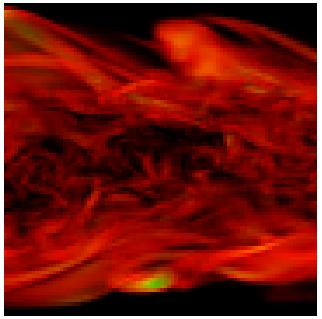
1) Luebke, D. and Humphreys, G.: How GPUs Work, *Computer*, Vol.40, No.2, pp. 96–100 (2007).
2) Nagayasu, D., Ino, F. and Hagihara, K.: A Decompression Pipeline for Accelerating Out-of-Core Volume Rendering of Time-Varying Data, *Computers and Graphics*, Vol.32, No.3, pp.350–362 (2008).
3) Du, Z., Chiang, Y.-J. and Shen, H.-W.: Out-of-Core Volume Rendering for Time-Varying Fields Using a Space-Partitioning Time (SPT) Tree, *Proc. 2nd IEEE Pacific Visualization Symp.*, pp.73–80 (2009).
4) NVIDIA Corporation: CUDA Programming Guide Version 3.0 (2010).
5) OpenGL Extension Registry: GL_NV_texture_ compression_vtc (2004). `http://oss.sgi.com/projects/ogl-sample/registry/NV/texture_compression_vtc.txt`.
6) Chen, J.: Turbulent Combustion Simulation (2011). `http://vis.cs.ucdavis.edu/Ultravis/datasets/`.
7) Ino, F., Tanaka, Y., Kitaoka, H. and Hagihara, K.: Performance Study of Non-rigid Registration Algorithm for Investigating Lung Disease on Clusters, *Proc. 6th Int'l Conf. Parallel and Distributed Computing, Applications and Technology (PDCAT'05)*, pp.820–825 (2005).



(a) 提案手法 $\langle 4,3 \rangle$　(b) 提案手法 $\langle 4,2 \rangle$



(c) 提案手法 $\langle 1,3 \rangle$（既存手法 PVTC）　(d) 提案手法 $\langle 1,2 \rangle$



(e) 非圧縮

図 4 乱流のレンダリング結果