2003－ＣＧ－113　(7)
2003／11／25

デスクトップ・ボリュームレンダリングのための解像度制御によるビデオメモリ使用量削減

池田孝利† 大西史泰†
伊野文彦† 萩原兼一†

本稿では，大規模なボリューム（例えば 512^3 ボクセル）を高速にボリュームレンダリング（VR）することを目標として，テクスチャを用いた VR におけるビデオメモリ使用量を削減する手法を提案する．提案手法は，ボリュームの断面（スライス）ごとの解像度を制御することにより，視点およびスライスの距離に応じてスライスの詳細度（LOD：Level of Detail）を適切に選択する．このとき，医療診断における応用を考慮し，解像度を制御しない場合と同一の VR 結果を得ることができる解像度を選択する．このように，スライスごとに適切な解像度を選択することで，画質を低下することなくビデオメモリ使用量の削減を実現し，高速かつ大規模な VR を目指す．

Reducing Video Memory Usage for Desktop Volume Rendering Using Resolution Control Mechanism

Takatoshi Ikeda,† Fumihiro Ohnishi,† Fumihiko Ino† and Kenichi Hagihara†

In this paper, we propose a method for reducing video memory usage for texture based volume rendering, aiming to enable high-speed volume rendering for large-scale volume datasets, for example, 512^3 voxel data. Our method controls the resolution of image slices so that realizes LOD (level of detail) switching according to the distance between the viewpoint and the slice. In this LOD switching, to consider the practical usage in clinical diagnosis, it selects an appropriate resolution that generates the identical image rendered without LOD switching. Thus, by selecting the appropriate resolution for each slice, we reduce the video memory usage so that realize high-speed and large-scale volume rendering without degrading the quality of rendered images.

1. はじめに

医用画像やシミュレーション結果などの 3 次元スカラデータ（ボリューム）の可視化は，データが持つ情報を空間的に理解するために有用である．ボリュームレンダリング（VR：Volume Rendering）[1] とは，ポリゴンなどの表面モデルを構築することなくこれらのデータから 3 次元画像を直接生成する手法である．しかし，VR は 3 次元データ数（ボクセル数）に比例した計算を伴うため，膨大な計算を実時間処理するための工夫が必要である．そこで，専用ハードウェア[2),3)] や並列計算[4)~6)] による高速化が提案されている．

一方，GPU（Graphics Processing Unit）の技術革新とともに，PC グラフィクスハードウェアの性能が急速に向上している．これらのハードウェアは，主にポリゴンレンダリングの高速化に主眼があるが，そのテクスチャ機能を VR に応用する手法[7),8)] が提案されている．例えば，2 次元テクスチャを用いた VR 手法では，座標軸に対して垂直な断面（スライス）を等間隔に並べたものをボリュームとみなし，テクスチャを張り付けたスライスを α 合成[8)] により半透明処理することで VR を模倣する．この VR 手法は安価な汎用ハードウェアにおいて高速な VR を実現できる．

しかし，グラフィクスカードが搭載するビデオメモリ上にボリュームの全体を保持できない場合，主記憶およびビデオメモリ間のデータ転送が発生し，処理速度が低下する問題がある．ゆえに，デスクトップ PC などの一般的な計算機環境において，大規模なボリュームを高速に処理できる VR 手法が必要である．

† 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

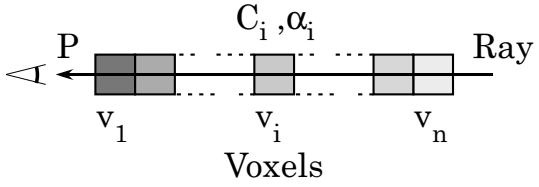

図 1 ボリュームの投影

本稿では，大規模なボリュームを高速に VR することを目的として，テクスチャに基づく VR のためのビデオメモリ使用量を削減する手法を提案する．提案手法では，スライスごとの解像度を制御することにより，視点とスライスとの距離に応じてスライスの詳細度（LOD：Level of Detail）を適切に選択する．このとき，結果画像における画質の低下が無視できない医療診断における応用を考慮し，解像度を制御しない VR の結果画像と同一の結果画像を得ることができる解像度を選択する．このように，スライスごとに適切な解像度を選択することで，ビデオメモリ使用量の削減を実現する．

以降では，提案手法の基礎となる技術について述べ（2 章），提案手法について述べる（3 章）．その後，本手法を適用した際に期待できる効果についての評価結果を示し（4 章），最後にまとめを述べる（5 章）．

2. ボリュームレンダリング（VR）

2.1 VR の原理

VR では，3 次元微小画素（ボクセル）の集合として量子化された空間を，光路に沿って通過したそれぞれのボクセルの持つ色と不透明度の累積演算を行い，投影面上の画素値 P を得る（**図 1**）．P の値は，視線が手前から i 番目に通過したボクセル v_i の持つ色を C_i，不透明度を α_i として，式 (1) で与える．

$$P = \sum_{i=1}^{n} \alpha_i C_i \prod_{j=1}^{i-1} (1 - \alpha_j) \qquad (1)$$

2.2 テクスチャによる VR

式 (1) は，任意のボクセル v_i における色 C_i が，v_i より視点側のすべてのボクセル $v_1 \sim v_{i-1}$ の不透明度を経て投影されることを表す．通過したボクセルのそれぞれにおける透過状態に着目すれば，あるボクセル v_i では，そのボクセルより奥側のボクセル v_{i+1} での透過光と v_i の色 C_i に不透明度 α_i を与えて，手前の位置のボクセルへと伝播している．$C_i^{'}$ を v_i までの累積値とすると，次の漸化式で表せる．

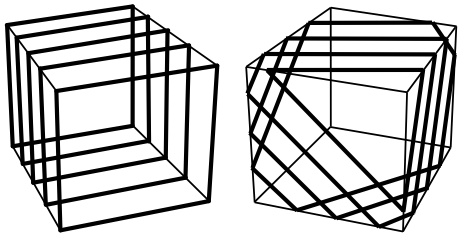

図 2 (a) オブジェクト整列スライス (b) 視野整列スライス

$$C_i^{'} = \alpha_i C_i + (1 - \alpha_i) C_{i+1}^{'} \quad (i = 1, 2, \ldots) \qquad (2)$$

このとき，投影面上の画素値 P は $P = C_1^{'}$ となる．

式 (2) は，累積演算処理において，α 合成と呼ばれる 3 次元空間内に配置する半透明描画指定された 2 次元平面テクスチャを評価処理する際の演算そのものである．したがって，α 合成指定された 2 次元平面テクスチャを視線と交差するように多層配置した空間のレンダリングにより，式 (1) と等価な結果が得られる[9]．

テクスチャにもとづく VR には，2 次元テクスチャによる手法と 3 次元テクスチャによる手法がある．

- 2 次元テクスチャによる手法（**図 2**(a)）：ボリュームを座標軸に対して垂直なスライスの集まりとみなし，それらのスライスにテクスチャを配置する．座標軸は 3 軸存在するため，スライスの向きは 3 種類考えられるが，視点からみたスライスの面積が最大となるスライスの向きを選択する．このように，テクスチャの位置および向きは座標軸に対して静的に定まり，視線の位置および向きに依存しない（オブジェクト整列スライス）．
- 3 次元テクスチャによる手法（図 2(b)）：ボリュームは 3 次元配列のデータとして一括して配置する（この 3 次元配列のデータを 3 次元テクスチャと呼ぶ）．その 3 次元配列のデータはそれぞれがボクセルに対応している．レンダリング時に処理対象とするボクセルは，視点に対して垂直（投影面と平行）になるような一連の平行面（スライス）に沿った位置のボクセルである．この一連のスライスの位置および向きは視点の位置に依存して動的に定まる（視野整列スライス）．

提案手法では，スライスごとに解像度を選択することが容易な 2 次元テクスチャによる VR 手法を用いる．3 次元テクスチャを用いる場合，座標軸に対して動的な位置のスライスに対し解像度を選択することは，一般に，市販のグラフィクスハードウェアでは実現できない．一方，2 次元テクスチャを用いる場合，各スラ

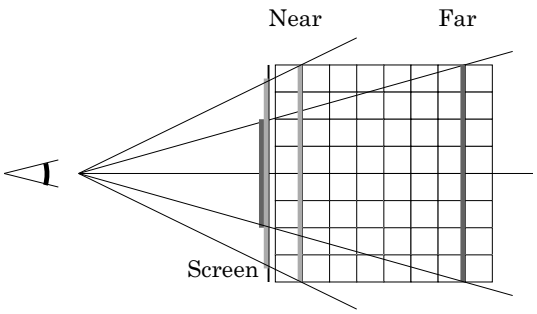

図 3 視点からテクスチャまでの距離と投影面上のテクスチャの大きさ

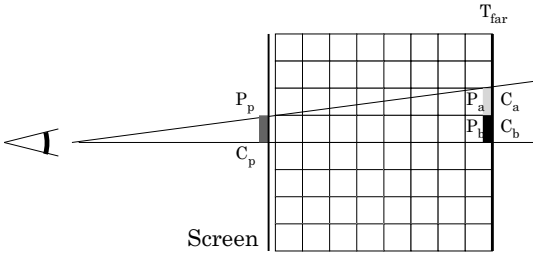

図 4 テクスチャの画素が投影面に投影される様子

イスの相対位置が静的に定まるため，スライスごとに異なる解像度を適用することが容易である．

2.3 詳細度（LOD）の制御

詳細度（LOD）の制御とは，主として高速化のために細部の描写の詳細度を調節することを指す．3 次元画像を構成する形状や画像を記述するデータを削減することで，画像生成のための計算量を削減し，高速化を図る．例えば，ゲームなどの実時間アプリケーションにおいて，膨大なポリゴンデータによって表現される巨大な 3 次元画像や，緻密なポリゴンにより表現された現実性の高い画像，テクスチャを多用した画像などの詳細度を制御することで，高速なレンダリングを実現する．

このように，精細な描画が不要な部分に対し，低解像度の描画演算処理することにより高速化が達成できる．

一方，高速化を目的としない，人間の視覚の特性を考慮して遠景にある物体や動いている物体などは低解像度で処理するなど，人間の視覚特性，初期視覚を考慮する研究も行われている[10]．

なお，既存の LOD 手法は処理結果における画質の低下を伴うが，提案手法では画質が低下しない解像度を適用する．

3. 提案する VR 手法

3.1 ビデオメモリ使用量の削減方針

ボクセルに基づく VR では，光線が透過するボクセルごとに演算が必要であるため，時間計算量はおよそ総ボクセル数に比例する．一方，テクスチャに基づく VR では，ボリュームをテクスチャデータとして扱うため，時間計算量はおよそテクスチャデータのサイズ（テクスチャのピクセル数の合計）に比例する．

また，ハードウェアによるレンダリングでは，演算が必要であるデータは，通常パイプライン構成の演算器に，ビデオメモリから転送される．しかし，データがビデオメモリ内にあらかじめない場合には，CPU がデータを主記憶からビデオメモリへ逐次転送する必要がある．ゆえに，テクスチャデータのサイズがビデオメモリの容量を上回る場合，描画時にデータ転送が頻発し処理速度が低下する．

ビデオメモリの容量を越えるテクスチャデータが必要なボリュームを VR する場合，使用するテクスチャデータをより小さくできれば，この速度低下を回避した VR が実現できる．使用するテクスチャデータを削減する手法として，以下の 4 種類が挙げられる．

(1) データ構造の低精度化：ビデオメモリが保持するデータ構造の内部表現を低精度なものに変更する．ビデオメモリの使用量を削減することができるが，アプリケーションによっては輝度・色の精度を下げた状態での使用は容易でなく，輝度・色の精度を下げられないなどの制限がある．

(2) データの圧縮：テクスチャデータを圧縮することにより，ビデオメモリの使用量を削減する．データ圧縮は，VR のみならずポリゴンレンダリングにおいても需要がある．

(3) 適応的なデータ構造の採用：同一の値を持ち，互いに隣接するボクセルを 1 個に集約することで，データ量を削減する．この際，値の誤差を許し，同一値とみなせるボクセルを集約する手法もある[11]．

(4) 解像度の制御：VR 処理時に必要な解像度まで低解像度化し，データを削減する．

本稿では，最終的に得られる結果を変化させないことを優先に考え，手法（4）について提案する．

3 次元空間に配置したテクスチャを投影面に投影するとき，視点からの距離が遠くなるほど，投影面上での大きさは小さくなる（**図 3**）．視点から遠いテクスチャT_{far} の画素 P_a，P_b が投影面の画素 P_p に投影される様子を**図 4** に示す．図のように，投影面上では視点に遠いテクスチャほど投影後の画素が小さいため，テクスチャの複数の画素が投影面の 1 画素に対応する

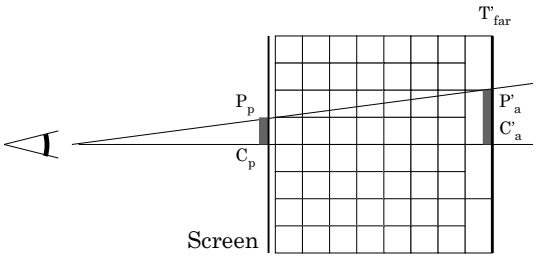

図 5 遠くのテクスチャを低解像度化した様子

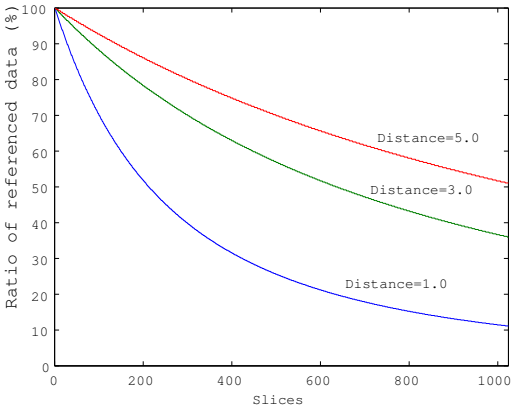

図 6 投影面の 1 画素に対応するデータの比率

ことがある．画素 P_a，P_b の持つ色 C_a，C_b はそれぞれ独立であるが，投影後は P_p の持つ 1 つの色 C_p に対応する．図 4 で示すように，T_{far} を低解像度化したテクスチャ$T_{far}^{'}$ をにおいて，C_p が変化しないような，P_a，P_b に対応する低解像度化された点 $P_a^{'}$ に色 $C_a^{'}$ を与えることができる．このような状況では，低解像化によって削減可能な画素データが発生していると考えられ，無駄な画素データを含んでいるといえる．

VR 時の投影面の 1 画素の大きさは視点とテクスチャの距離に依存して相対的に変化するため，そのような無駄なテクスチャの画素（ボリューム中のボクセルに対応）の発生は，幾何演算により事前に予想できる．3 次元座標中の座標 $(-1,-1,-1)$〜$(1,1,1)$ の 1024^3 ボクセルからなるボリュームを 1024 枚の 2 次元テクスチャにより VR 処理する場合の投影後の 1 画素に対応するデータについて予測を行った．ボリュームと視点間の距離を 1，3，5，投影面の解像度とスライスの解像度は同一という条件を設定した．横軸は視点に近い方から昇順に番号を振った場合のスライス番号，縦軸はそのスライス上のデータ中の VR 演算時に投影面の 1 画素に対応する画素の比率をとったグラフを **図 6** に示す．この 図より，視点から離れたスライスほど

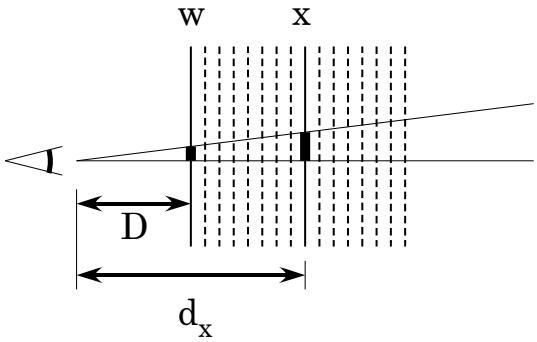

図 7 投影画素

参照されるデータが減っている様子がわかる．このような無駄なボクセルに相当するテクスチャデータを抑えるための解像度は，視点，投影面，ボリュームの位置関係から導出可能なものである．さらに，この解像度は対象となるボリュームデータとは無関係であるため，VR 処理実行以前に解像度を選択し，その解像度のテクスチャデータを用意することが可能である．

3.2 テクスチャの解像度の制御方法

一般に，市販のグラフィクスハードウェアでは，使用できるテクスチャの解像度に制限があり、任意の解像度を選択できないことが多い．2 の巾乗に制限されている．そこで，より多くのグラフィクスカードにおける応用を目的として，提案手法では 2 次元テクスチャの解像度を 2 の巾乗で与えるものとする．すなわち，適切な詳細度の選択は 2 の巾乗の解像度に基づく．

以降では，視点，投影面 w およびスライス x が与えられたとき，x に対する適切な解像度 R_x を選択する手法について述べる．簡単のために，ボリュームを構成するスライスのうち，視点に最も近いものが w と一致するものとする（**図 7**）．視点から w までの距離を D，視点から制御対象となる x までの距離を d_x とすると，x における画素は w において D/d_x に縮小投影される．したがって，x は w に対して d_x/D の解像度であれば，解像度制御なしのときの VR 結果を損なうことなく解像度を減少できる．ここで，詳細度の選択は 2 の巾乗で行うことを考慮し，d_x/D をの 2 の巾乗へ丸める必要がある．解像度を減少する際，それまで参照されていた画素が失われないことを前提条件とすると，R_x は次式で与えることができる．

$$R_x = R_{org} \cdot \frac{1}{2^{\lfloor log_2(\frac{d_x}{D}) \rfloor}} \tag{3}$$

ここで，R_{org} は制御前の x の解像度を表す．

本手法には，VR 結果が，データ削減により変化しないという利点がある．加えて，視点などの位置関係

表 1 実験環境

CPU	Pentium4 2.53GHz
Main memory	768MB
AGP	4x
Graphics card	(1) nVIDIA GeForce FX5900Ultra w/ 256MB Video memory
	(2) nVIDIA GeForce FX5600 w/ 128MB Video memory
OS	RedHat Linux 9
Rendering API	OpenGL（Mesa）

のみに依存した手法であるため，VR 処理する対象のボリュームデータに依存しない性能改善が得られる．ボクセルの持つ色や不透明度に依存しないため，解像度の 3 乗オーダである各ボクセルを処理対象とせず計算量が少ないという利点がある．

4. 予 備 実 験

提案手法による改善効果を明らかにするために予備実験を行った．実験では，以下の 2 点について提案手法の改善効果を確認した．

- 理論的な解析によるビデオメモリ使用量の削減
- 期待できる VR 性能の予測

表 1 に実験で用いた計算機の主な仕様を示す．ビデオメモリ容量の異なるグラフィクスカードを 2 種類用意し，各々を使い分けて性能を計測した．

4.1 ビデオメモリ使用量の比較

1024^3 ボクセルのボリュームを座標 $(-1,-1,-1)$〜$(1,1,1)$ に配置し，そのボリュームを距離 D から VR すると仮定する．**表 2** に，式 (3) に基づいて解像度を制御する提案手法および解像度を制御しない手法が要するメモリ使用量 U_1 および U_2 を示す．ここで，ボクセル 1 個を保持するために 4 バイトを要するものとして計算した．

D の値が増大するとともに U_1 の値も増大していることから，視点がボリュームに近いほど提案手法によりメモリ使用量を削減できていることが分かる．この理由は，視点およびボリューム間の距離が短い場合，視野角が広がり視点から離れた位置のテクスチャに対して，より低解像度を選択してもよいからである．一方，提案手法を用いない場合，U_2 は D の値と関係なく一定となる．

このように，視点およびボリューム間の距離が短い場合，提案手法の解像度制御が効果的に働くことを理論的に示すことができた．

4.2 本手法適用による性能改善性

本手法適用により VR 処理性能が改善されるかを推測するために，2 次元テクスチャを使用する方法で

表 2 ビデオメモリ使用量の比較

距離	メモリ使用量（MB）		削減率（%）
D	U_1：制御あり	U_2：制御なし	σ^*
0.5	1,600	4,096	61
0.75	2,176	4,096	47
1.0	2,560	4,096	37
1.25	2,944	4,096	28
1.5	3,328	4,096	19

$*: \sigma = 100 \cdot (1 - U_1/U_2)$

VR の処理速度を測定した．実験では，VR 対象とするボリュームデータのサイズを変化させ，グラフィクスカードに実装されたビデオメモリの容量を超える前と後での VR 処理速度の変化を確認した．対象ボリュームのサイズは 512^3 ボクセルで，このボリュームから解像度 512^2 のピクセルの 2 次元テクスチャを生成した．テクスチャメモリ使用量を変化させるにあたっては，2 次元テクスチャの解像度は変化させず，スライス数を変化させている．それぞれのグラフィクスボードで，解像度制御を行っていない VR の処理速度データを **図 8**(a)，図 8(b) に示す．どちらもテクスチャデータ量がグラフィクスハードウェアに搭載されているビデオメモリの容量を超えた段階で処理速度が落ちている様子が表れている．

本手法適用時には，表 2 に示された比率でのメモリ使用量の減少状態となるため，図に表される処理速度特性の変異点は，手法適用前のメモリ使用量が増大する方向へ移動する．この変異点の移動の結果，高速に処理できる範囲が拡大する．例として表 2 中の距離 $D = 1.0$ の場合を 図 8(c) で示す．

表 1 に示した実験環境でグラフィクスカード（1）の本手法を適用しない場合の VR 処理速度は，400 スライス時にテクスチャメモリ使用量 400MB に対して 1 フレームあたり 323 ms であり，250 スライス時にはテクスチャメモリ使用量 250MB に対して 57 ms である，本手法適用時には，解像度制御によりビデオメモリ使用量が 37% 削減され 252MB（= 400MB ·0.63）となる．これはビデオメモリ容量の 256MB を下回っているため速度低下前の処理性能を適用し，処理速度は 57 ms /250 スライス ·400 スライス = 91 ms と予想できる．この場合の速度向上比は，323 ms /91 ms = 3.5 である．また，$512 \times 512 \times 400$ ボクセルのボリュームの VR について，解像度制御の適用前は毎秒 3.1 フレームであったが，適用後では毎秒 11.0 フレーム程度の処理速度が期待できる．

5. ま と め

本稿で提案した手法は，視点などの位置関係によっ

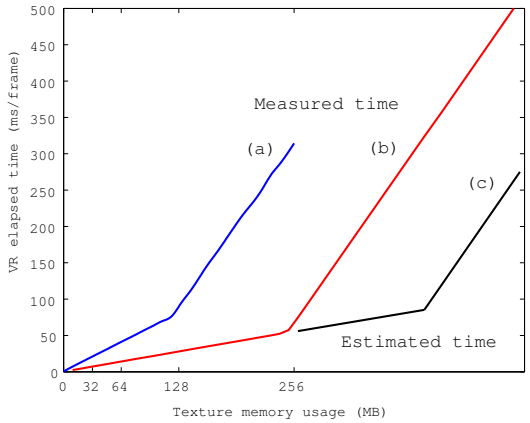

図 8 VR に要する時間 ((a))FX5600：ビデオメモリ容量 128MB) 1MB/slice,(bFX5900Ultra：ビデオメモリ容量 256MB) 1MB/slice，(c) 性能改善後予測 FX5900Ultra(256MB) 1MB/slice

てビデオメモリ使用量を効率良く削減できるため，削減量に応じた高速化が期待できる．大きな削減効果が期待できない場合でも，ビデオメモリはハードウェアに固定的に実装されていることが多く増設が容易でない，そのため搭載されているビデオメモリを効率的に使用することは，有用であると考えられる．本研究では，解像度制御を適用する対象をそれぞれのスライスについて独立に行ったが、より柔軟に解像度の制御を行うことが今後の課題である．

謝辞 本研究の一部は，日本学術振興会未来開拓学術研究推進事業（JSPS-RFTF99I00903），平成 15 年度厚生労働省がん研究助成金（課題番号 15 指-1），科学研究費補助金基盤研究 (C)(2)(14580374) および若手研究 (B)(15700030) の補助による．

参 考 文 献

1) Akenine-Möller, T. and Haines, E.(eds.): *Real-Time Rendering*, AK Peters, Ltd., Natick, MA, second edition (2002).
2) Pfister, H., Hardenbergh, J., Knittel, J., Lauer, H. and Seiler, L.: The VolumePro Real-Time Ray-casting System, *Proc. SIGGRAPH '99*, pp. 251–260 (1999).
3) 緒方正人, 村木茂, マクワンリュウ, 劉学振: 高並列ボリュームレンダリングを可能にするパイプライン画像重畳装置の設計と評価, 電子情報通信学会論文誌, Vol. J86-D-II, No. 2, pp. 290–301 (2003).
4) Muraki, S., Ogata, M., Ma, K.-L., Koshizuka, K., Kajihara, K., Liu, X., Nagano, Y. and Shimokawa, K.: Next-Generation Visual Supercomputing using PC Clusters with Volume Graphics Hardware Devices, *Proc. High Performance Networking and Computing Conf. (SC2001)* (2001).
5) 丸山悠樹, 中田智史, 高山征大, 津邑公暁, 五島正裕, 森眞一郎, 富田眞治: 汎用グラフィクスハードウェアを用いた並列ボリュームレンダリングの実装, 情報処理学会研究報告, 2003-ARC-154, pp. 61–66 (2003).
6) Takeuchi, A., Ino, F. and Hagihara, K.: An Improved Binary-Swap Compositing for Sort-Last Parallel Rendering on Distributed Memory Multiprocessors, *Parallel Computing*, Vol. 29, No. 11/12, pp. 1745–1762 (2003).
7) Akeley, K.: RealityEngine Graphics, *Proc. SIGGRAPH '93*, pp. 109–116 (1993).
8) Brady, M., Jung, K., Nguyen, H. T. and Nguyen, T.: Two-Phase Perspective Ray Casting for Interactive Volume Navigation, *Proc. 8th IEEE Conf. on Visualization*, pp. 183–190 (1997).
9) Rezk-Salama, C., Engel, K., Bauer, M., Greiner, G. and Ertl, T.: Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization, *Proc. SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware (HWWS'00)*, pp. 109–118 (2000).
10) Reddy, M.: Perceptually Optimized 3D Graphics, *IEEE Computer Graphics and Applications*, Vol. 21, No. 5, pp. 68–75 (2001).
11) 山崎俊太郎, 加瀬究, 池内克史: PC グラフィックスハードウェアを利用した高精度・高速ボリュームレンダリング手法, 情報処理学会研究報告, 2001-CVIM-113, pp. 71–78 (2001).