計算機アーキテクチャ　128－2
ハイパフォーマンス
コンピューティング　70－2
（1998．3．5）

階層格子ボリュームデータの実時間可視化

藤原雅宏† 五島正裕†
森眞一郎† 富田眞治†

本稿では，ボリュームスライス分割方式のボリュームレンダリング専用並列計算機における階層格子ボリュームデータの可視化手法について述べる．一般に，レイキャスティング法によるボリュームレンダリングは，ボリュームデータへの不連続なアクセスを伴い，その高速化を妨げている．そこで，視線のボリュームスライスへの入射順序を制御した上で，ボクセルデータへのアクセス機構を付加することにより，構造化されたボリュームスライスからの高速なボクセルデータの取得を可能とする手法を示した．

Realtime Visualization Method for Hierarchical Grid Volume Data

Masahiro Fujihara,† Masahiro Goshima,† Shin-ichiro Mori†
and Shinji Tomita†

In this paper, we describe a visualization method for hierarchical grid volume data on the volume-slice type volume rendering machine. In general the ray-casting algorithm causes random memory access on the volume-slice, and it prevents high-speed processing. In order to solve this problem, we proposed the high-speed access method to the structured volume-slice by controlling ray penetration order and additional access mechanism.

1. はじめに

ボリュームレンダリング法は，3次元データの内部構造の可視化が可能であるため，医療画像の生成や科学技術計算結果の解析のための有力な手法として注目されている．

しかしながら，ボリュームレンダリングを行うための計算量は大きく，高速な可視化は困難である．そのため，従来から *ReVolver*/C40[1]やCUBE[2]などボリュームレンダリングに特化したさまざまな専用並列計算機の開発が行われてきた．これらの専用並列計算機では，立方体格子上でサンプリングされたボリュームデータを対象とした実時間での可視化処理を実現している．

一方，近年のCTやMRIなどに代表される3次元医療計測機器の進歩により，1024^3ボクセルの高解像度ボリュームデータを取得することが可能になるなど[3]，可視化すべきボリュームデータの大きさは増大する傾向にあり，それに伴いデータの取り扱いが困難になりつつある．

そのような巨大なボリュームデータを効率的に扱うために，サンプリング格子の密度を適応的に変えることにより，ボリューム空間をその重用度に応じた解像度で表現することができる階層格子構造は有力なボリュームデータの表現手法とされている[4]．

本稿では，ボリューム空間を**ボリュームスライス**と呼ばれる2次元平面の集合として扱うボリュームスライス分割方式のボリュームレンダリング専用並列計算機上で，レイキャスティング法を用いた階層格子ボリュームデータの可視化手法を提案する．

以下，2章では，ボリュームスライスの格納に用いるデータ構造とその特性について述べ，3章で，ボクセルデータへのアクセス順序を制御するための視線入射順制御の提案を行い，4章において，視線入射順制御に基づいて効率的なボクセルデータへのアクセスを行うボクセルデータアクセス機構の提案を行い，5章において，提案手法の有効性を医療画像を用いて評価する．最後に6章において本論文をまとめる．

2. ボリュームスライス分割とその表現

ボリュームスライス分割方式のボリュームレンダリング専用並列計算機では，各ボリュームスライスにプロセッサエレメントを割り当てることによりスライス間の並列性を抽出し，高速に可視化処理を行うことができる[1]．この方式をとる計算機のモデルを図1に

† 京都大学大学院工学研究科情報工学教室
Department of Information Science, Graduate School of Engineering, Kyoto University

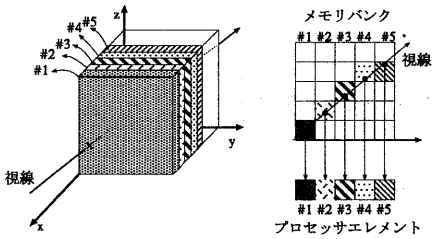


図1　ボリュームスライス分割方式

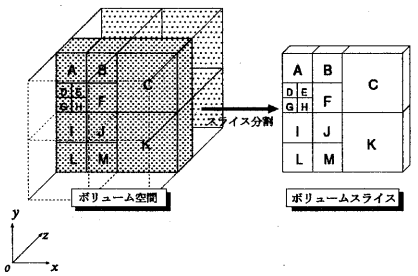


図2　Octree 構造のボリュームスライス分割

示す.

2.1　ボリュームデータ

階層格子ボリュームデータは，解像度が異なる複数の立方体格子系により構成される．代表的なものに**octree 構造**があり，この構造は空間の 8 等分割を繰り返すことにより構成され，空間中のデータの分布に応じて格子の密度を変えることができる[5]．

2.2　ボリュームスライス分割

ボリュームスライス分割方式では，3 次元のボリュームデータから 2 次元のボリュームスライスを抽出する必要がある．

直方体格子で構成されるボリュームデータを x 軸に垂直な平面でスライス分割する場合，ボリューム空間内の座標 (x, y, z) に位置するボクセルデータは，n 番目のボリュームスライス上の座標 (n, y, z) に格納される．このスライス分割を y, z 軸にも適用することで，任意の方向からの視線に対して可視化処理を行うことができる．また，階層格子構造を持つボリュームデータの場合においても，格子構造に応じたスライス分割処理を行うことにより，ボリュームスライスを抽出することができる．

以後の議論では，図 2のように octree 構造を持つボリュームデータから抽出されたボリュームスライスについて述べる．

2.3　ボリュームスライスの表現

従来のボリュームレンダリング専用並列計算機では，抽出されたボリュームスライスは正方格子に分割され，2 次元配列に格納されてきた．しかしながら，この手法ではボリュームデータ量が増大してしまい階層格子構造で表現された巨大なデータを格納することは困難となる．従って，本稿では**ランレングス構造**および**quad-tree 構造**を用いて，ボリュームスライスを構成することによりデータサイズの減少を図る．

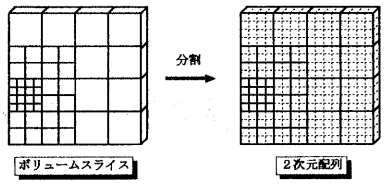


図3　2 次元配列構造への分割

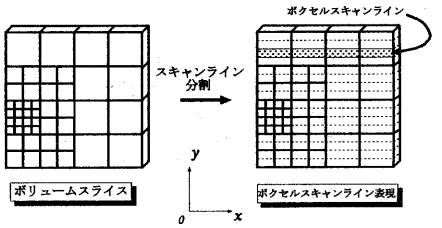


図4　ボクセルスキャンラインへの分割

2.3.1　2 次元配列構造

2 次元配列構造は，ボリュームスライスを表現する最も基本的な構造である．図 3のように，階層格子を構成する最も密度の高い格子でボリュームスライスを等分割することにより構成される．本稿では，この格子の大きさを**最密格子サイズ**と呼ぶ．

一般に，2 次元配列構造はメモリ上の連続したアドレス空間に格納される．従って，配列の大きさが決まればボクセルデータの格納アドレスも決まるため，スライス上の任意の座標のボクセルデータへの高速なアクセスが可能である．

2.3.2　ランレングス構造

ランレングス構造は，ボクセルデータと同一ボクセルデータが連続して現れる長さである**ラン長**からなる**ラン**の集合である[6]．本稿では図 4のようにボリュームスライスを最密格子サイズで x 軸方向に分割する．この分割されたデータを**ボクセルスキャンライン**と呼び，ランレングス構造を用いてメモリ上に格納する．全ボクセルスキャンラインをランレングス構造で表現することでボリュームスライスを構成する (図 5).

ランレングス構造で表されたボクセルデータへのアクセスには，ボクセルスキャンライン上での順次アクセスが向いており，ボクセルデータに対するランダムアクセスが行われた場合は，目的のボクセルに到達するために，ボクセルの x 座標値に比例した数のランを読み飛ばす必要があるため，効率が悪くなる．

2.3.3　quad-tree 構造

quad-tree 構造は，図 6のようにボリュームスライスを 4 分割し，ボクセルデータを木の節点の各要素に割り当て，さらに分割が必要な領域に関しては 4 分割

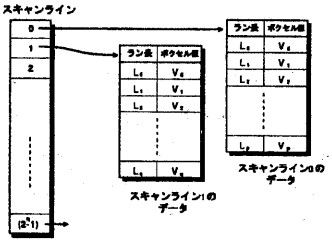


図5　ランレングス構造

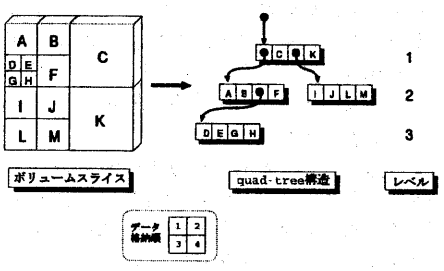


図6　ボリュームスライスの quad-tree 表現．

を繰り返すことにより平面を構成する構造である[5]．本稿では，quad-tree 構造においてスライスの分割回数を**レベル**と呼び，最上位層から順にレベル 1, 2, ... , n と定義する (図 6)．また，各節点は 4 つの要素を持ち，下位レベルの節点へのリンクもしくはボクセルデータを格納する．

quad-tree 構造は，親子，兄弟関係にある節点へのアクセスには向いているが，ランダムアクセスには不向きであり，一般に木の高さに比例した時間を要する．

3. 視線入射順制御

レイキャスティング法でボリュームレンダリングを行う場合，スクリーン上の各ピクセルに対して，視点とピクセルをつらぬく**視線ベクトル**が通るボクセルデータを各ボリュームスライスから読み出す処理 (**サンプリング**と呼ぶ) が必要である．しかしながら，レイキャスティング法により生じた視線ベクトルは一般に視点，ピクセルおよびボリューム空間の位置関係により変化するため，ボリュームスライス上の各データ構造に対するランダムアクセスが生じる．

2 次元配列構造では，ランダムアクセスに対しても高速なボクセルデータへのアクセスが可能であるが，ランレングス構造および quad-tree 構造においては，2.3節で述べたように各データ構造に適応した順序でのみ高速なデータへのアクセスが可能である．そのため，これらの構造へのランダムアクセスが必要なレイキャスティング法では，処理速度の低下が避けられない．

本章では，視線のボリュームスライスへの入射順序を制御することにより，ボリュームスライス上をつらぬく視線が描く軌跡を常にボクセルスキャンラインに平行にすることで，高速化を図る手法を提案する．

3.1 視線ベクトルの生成

図 7のように，スクリーン座標における水平および垂直方向の単位ベクトルを，$\mathbf{S}_h$, $\mathbf{S}_v$ とし，視点からスクリーンの左上☆に向うベクトルを $\mathbf{S}_b$ とする．ここで，スクリーン座標が (s_x, s_y) であるピクセルに対する視線ベクトルを $\mathbf{R}$ とすると，$\mathbf{R}$ は次式で表される．

$$\mathbf{R} = \mathbf{S}_b + s_x\mathbf{S}_h + s_y\mathbf{S}_v$$

この視線生成処理をスクリーンのスキャンライン順に行うと，ボリュームスライスに任意方向の視線が入射することになる．

3.2 視線入射順制御

ボリュームデータは，図 7のように各座標軸に垂直に置かれており，ある座標軸 (図では z 軸) に垂直な平面によってボリュームスライスに分割されているとする．また，座標原点からスクリーンの左上へのベクトルを $\mathbf{S}_b = (b_x, b_y, b_z)$，スクリーンの水平方向の単位ベクトルを $\mathbf{S}_h = (h_x, h_y, h_z)$，スクリーンの垂直方向の単位ベクトルを $\mathbf{S}_v = (v_x, v_y, v_z)$ と定義する．

ここで，実際のスクリーンに対し，仮想スクリーンを想定する．仮想スクリーン上の座標 (p, q) に対して q番目のスキャンラインに透視投影されるボクセルは，次の写像 fによりボリュームスライス上で x 軸に平行な軌跡を持つボクセルスキャンラインを生成する．

$$f : \begin{pmatrix} S_x \\ S_y \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ k_z & -1 \end{pmatrix} \begin{pmatrix} \mathrm{p} \\ \mathrm{q} \end{pmatrix}$$

ここで k_zは，

$$k_z = \begin{vmatrix} h_y & h_z \\ b_y & b_z \end{vmatrix} \Big/ \begin{vmatrix} v_y & v_z \\ b_y & b_z \end{vmatrix}$$

とする．

従って，仮想スクリーン上のスキャンラインにそって視線を生成することにより，任意のボリュームスライスに対して常にボクセルスキャンライン方向に視線を入射することができる．

4. ボクセルデータアクセス機構

視線入射順制御を行うことにより，ボリュームスライスに対する x 軸方向の空間的なアクセスの連続性を仮定することができる．本章ではこの連続性を利用し，プロセッサとボリュームメモリとの間に，ボクセルデータアクセス機構を付加することにより，高速なボクセルデータのサンプリングを可能にする手法について提案する．

4.1 ランレングス構造からのサンプリング

アクセス連続性の仮定より，直前のサンプリング時に求めたランの絶対座標を記憶しておき，新たにサン

☆ スクリーンの左上を，スクリーン座標における原点とする．

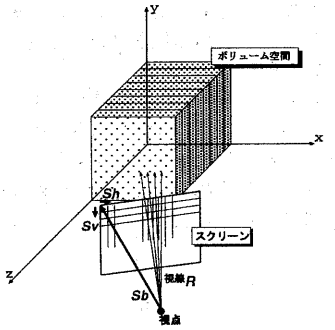


図7　透視投影法によるレイキャスティング

プリングするデータの絶対座標と比較することにより，ボクセルデータ読み出しに要するメモリアクセス回数の削減が期待できる．

4.1.1　サンプリング方式

直前のサンプリングを行った後のランの位置を示すX_{pos}と，そのランが占める領域を表すx座標の上限値X_{max}をそれぞれ記憶しておき，新たにサンプリングするボクセルの座標を(X, Y)とするとき，次のようにボクセルデータの取得を行う．

- $X \leq X_{max}$のとき
  X_{pos}のランが持つボクセルデータが求めるボクセルデータである．
- $X > X_{max}$のとき
  X_{pos}を次のランの位置に移動後，新たなランをメモリから読み出し，X_{max}に新たに読み出したラン長を加え，$X \leq X_{max}$となるまで上記の処理を繰り返し行う．

レイキャスティング法では通常，任意のボクセルスキャンラインに対して視線が到達するためX_{pos}, X_{max}を保持するための記憶領域が2^n個必要となるが，視線の入射順序を制御することにより，1個に抑えることができる．従って，ハードウェア化を考慮した場合，大幅なハードウェア量の削減につながる．

4.1.2　メモリアクセス特性

各スキャンラインで最初にボクセルデータを取得する際に，x座標値に比例した回数のメモリアクセスが必要である．

2度目以降は，1度目のサンプリング時に得た情報を基に相対的にボクセルデータが検索される．このとき，ボリュームメモリへの平均アクセス回数は次のようになる．

- 視線の入射間隔$\Delta R \leq$最密格子サイズΔgのとき
  視点がボリューム空間に近付いているときや，ピクセルの解像度がボクセルの解像度に比べて高いときに生じる．平行投影法の場合は，ボクセルのサンプリングは$\Delta g/\Delta R$回に1度メモリへのアクセスを必要とする．
- 視線の入射間隔$\Delta R >$最密格子サイズΔgのとき
  ボリューム空間を遠方から眺めているときや，ボクセルの解像度がピクセルに対して高いときに生じる．1度のサンプリングで$\Delta R/\Delta g$個のボクセルを読み飛ばす必要がある．従って，平均ラン長をLとするとき，平均$\frac{\Delta R/\Delta g}{L}$回のメモリアクセスが必要となる．

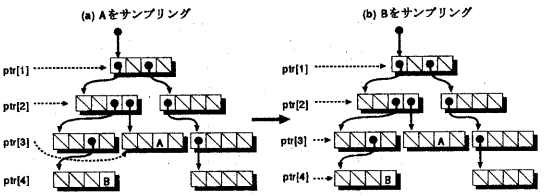


図8　quad-tree構造からのサンプリング

4.2　quad-tree構造からのサンプリング

図8のように，サンプリングの際にquad-treeの各レベルで訪れた節点へのポインタを記憶しておき，その情報から新たにサンプリングするボクセルデータが存在するアドレスを求めることによりメモリアクセス回数の削減を図る．

4.2.1　サンプリング方式

直前にサンプリングしたボクセルのレベルをV_L，座標を(x, y)，新たにサンプリングするボクセルの座標を(X, Y)，ボリューム空間の1辺の大きさを2^nとする．ボクセルのX座標値について最上位bitからの連続した一致bit数を算出し，k bitまで一致した場合，次のようにボクセルデータの取得を行う．

- $k \geq V_L$の場合
  前回サンプリングした点と，新たにサンプリングする点は，同じボクセルデータを持つので，新たに読み出す必要はない．
- $k < V_L$の場合
  前回のサンプリング時に記憶していたレベル$(k+1)$の節点へのポインタからquad-treeを探索することで，新しいサンプリング座標のボクセルデータを得ることができる．

4.2.2　メモリアクセス特性

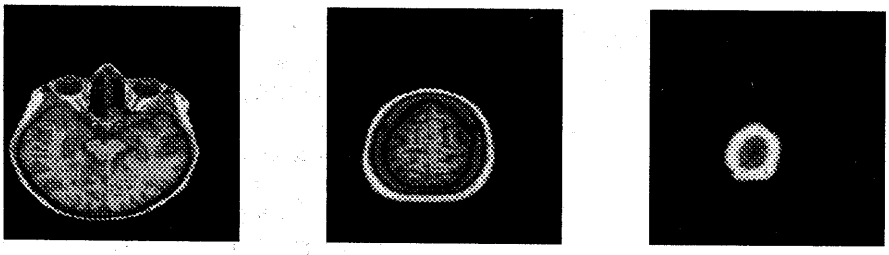
本手法でボクセルデータを取得する場合，次のメモリアクセス回数で，ボクセルデータを取得することができる．

$$max(\ (\textbf{データのレベル} - k),\ 0)\ \text{回}$$

ここで**データのレベル**は，新たにサンプリングするデータが格納されているquad-tree上でのレベルである．

5.　提案手法の評価

本章では，**2次元配列構造**，**ランレングス構造**，**quad-tree構造**の3つのデータ構造について，ボリュームスライスを表現するために必要な**データサイズ**，



(a) 高密度　　(b) 中密度　　(c) 低密度

図 9　評価に用いたボリュームスライス

ボリュームデータの密度	データ構造	要素数 (個)	要素あたりのサイズ (*byte*/個)	管理領域サイズ (*byte*)	合計データサイズ (*byte*)
(a) 高密度	2次元配列構造	65536	1	2	65538
	ランレングス構造	19201	2	514	38916
	quad-tree 構造	8382	8	2	67058
(b) 中密度	2次元配列構造	65536	1	2	65538
	ランレングス構造	11725	2	514	23964
	quad-tree 構造	4956	8	2	39650
(c) 低密度	2次元配列構造	65536	1	2	65538
	ランレングス構造	2937	2	514	6388
	quad-tree 構造	1162	8	2	9298

表 1　ボリュームスライスのデータ量

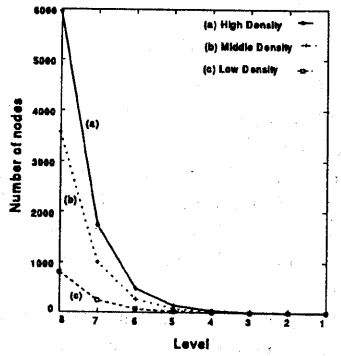


図 10　quad-tree レベル別節点数

ボクセルデータを取得するために要するメモリアクセス回数という観点から，人間の頭部のボリュームデータ☆ を用いた評価を行うことによりその有効性を確かめる．また，評価にはボリュームデータに占める物体の割合が異なる3つのボリュームスライスを用いた (図 9)．

5.1　データサイズ

図 9に示すボリュームスライスを，各構造で表現するために要したデータサイズを表 1 ☆☆ に示す．

☆ 256^3ボクセル，256 階調

☆☆ 表中の要素あたりのサイズは，ボクセルデータ格納サイズ，ラン長格納サイズ，ポインタ格納サイズをそれぞれ 1, 1, 2 *byte* として算出した．

ランレングス構造による圧縮の結果，2次元配列構造に対して密度により 9.7%〜59.4%のデータサイズで格納された．一方，quad-tree 構造では 14.2%〜102.3%となり，高密度のボリュームスライスにおいてデータサイズが2次元配列構造を越える結果となった．これは，quad-tree 構造のデータサイズが，ポインタの格納に要するデータサイズに比例することが原因である．

しかしながら，図 10より quad-tree 構造の節点の大部分は最大レベルのボクセルデータであり，それらはもはや分割する必要がないので，ポインタを格納するための領域は必要なく，要素あたりのサイズを 4*byte* にすることができる．この点を考慮すると各密度における quad-tree 構造を格納するために必要なデータサイズは，それぞれ 43228, 25392, 6056*byte* となり2次元配列構造に対して 9.2%〜66.0%のデータサイズで格納することが可能である．

5.1.1　メモリアクセス回数の評価

密度の高い (図 9(a)) のボリュームスライスに入射する視線の間隔とサンプリングに要した平均メモリアクセス回数との関係を図 11に示す．

スクリーンとボリュームデータの解像度のバランスがとれる視線間隔 1 のときの階層格子へのアクセス回数は，ランレングス構造 (図 11(b)) においては平均 0.3 回となり，2次元配列構造 (図 11(a)) よりも高速化された．また，quad-tree 構造 (図 11(d)) の場合は平均 1.3 回となる．これは，一般的な運用状況において，従来の2次元配列を用いた実時間可視化システ

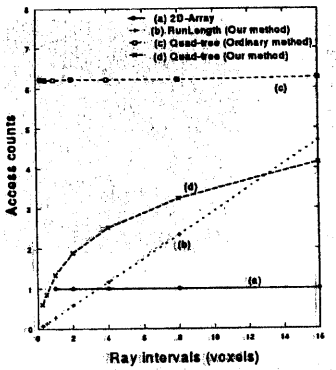


図 11　平均メモリアクセス回数

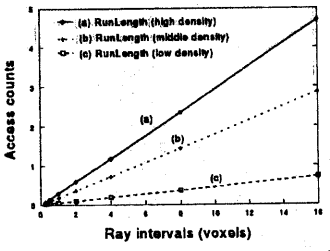


図 12　密度別平均メモリアクセス回数 (ランレングス構造)

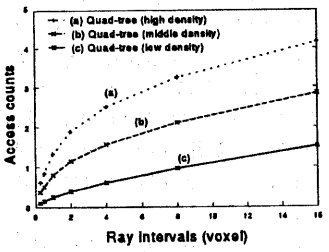


図 13　密度別平均メモリアクセス回数 (quad-tree 構造)

ムに本稿が提案する手法を用いて階層格子ボリュームデータの可視化を行った場合，十分な可視化速度を得ることができることを示している．

また，ランレングス構造 (図 11(b)) では視線間隔に比例する回数でアクセスが増加する．従って，視線間隔が広がるにつれ可視化速度も比例して低下してしまう．このことは，巨大なボリュームデータの全体像を観測する場合に問題を引き起こす．一方，quad-tree 構造では本稿で提案した手法により，視線間隔によるメモリアクセス回数の増加を抑えることができた (図 11(d))．

一方，図 12, 13 はボリュームデータの密度変化によるアクセス数の変化を示している．これらの図は局所的に高い密度を持つボリュームデータを効率的に可視化していることを表している．

6. ま と め

本稿では，ボリュームスライス分割方式のボリュームレンダリング専用並列計算機における，階層格子ボリュームデータの可視化手法について述べた．

ボリュームスライスを表現するためのデータ構造として，従来の 2 次元配列構造に加えてランレングス構造，quad-tree 構造の適用を行った．また，レイキャスティング法によりこれらの構造からボクセルデータを取得するためのサンプリング機構を設けた上で，視線入射順制御を行うことによりボクセルデータに高速にアクセスできることが明らかとなった．

また，ランレングス構造でボリュームスライスを保持した場合，その可視化速度は視線間隔の影響を受けやすい．そのため，ボリュームデータの狭い範囲を重点的に観測する場合には高速な可視化が行える反面，大域的な観測を行う場合には大幅な速度低下を伴う．一方，quad-tree 構造は視線間隔による影響を受けにくく，広い範囲の観測において実用的な可視化性能を得ることができた．

謝　辞

日頃よりご討論いただく京都大学大学院工学研究科情報工学教室富田研究室の諸氏に感謝致します．

参 考 文 献

1) 對馬雄次ほか: ボリュームレンダリング専用並列計算機 *ReVolver* のアーキテクチャ, 情報処理学会論文誌, Vol. 36, pp. 1709–1718 (1995).
2) Pfister, H., Kaufman, A. and Chiueh, T.: Cube-3: A Real-Time Architecture for High-Resolution Volume Visualization, *In 1994 Workshop on Volume Visualization*, pp. 75–83 (1994).
3) BOER, M. et al.: Evaluation of a Real-time Direct Volume Rendering System, *Computer & Graphics*, Vol. 21, No. 2, pp. 189–198 (1997).
4) Nielson, G. et al.: *Scientific Visualization*, IEEE Computer Society Press (1997).
5) Samet, H. and Webber, R.: Hierarchical Data Structures and Algorithms for Computer Graphics, *IEEE Computer Graphics & Applications*, pp. 48–68 (1988).
6) Montani, C. and Scopigno, R.: Rendering Volumetric Data using the STICKS Representation Scheme, *SIGGRAPH '90*, Vol. 24, No. 5, pp. 87–93 (1990).