芸術科学会論文誌 Vol.1 No.1 pp. 1-7

空間分割シアーワープ法を用いた並列ボリュームレンダリングシステム

緒方正人[†], 村木茂[††], クワンリューマ[†††],
梶原景範[†], 劉学振[†], 越塚健児[†]

† 三菱プレシジョン（株） †† 独立行政法人 産業技術総合研究所 ††† カリフォルニア大学デービス -

A Parallel Volume-Rendering System using Spatially Partitioned Shear-Warp Method

Masato Ogata[†], Shigeru Muraki[††], Kwan-Liu Ma[†††], Kagenori Kajihara[†], Liu Xuezhen[†] and Kenji Koshizuka[†]

† Mitsubishi Precision Co.,Ltd. †† National Institute of Advanced Industrial Science and Technology.
††† California State University of Davis.

Abstract 科学技術上の諸問題を計算機を用いて解き，結果を直感的に把握したいとの要求，あるいは高度医療における MRI, CT, 超音波診断装置などのデータの表示，などから大容量の３次元情報を可視化する高速かつ安価なボリュームレンダリングシステムが必要とされている．ボリュームの可視化は， 処理するデータ量が膨大であることから，高速表示には，高価なスーパコンピュータや専用ハードウエアが用いられている．本論文は，市販の PC を複数用いて構成する低価格な高速ボリュームレンダリングシステムを提案し， その性能評価を示す．このシステムの高速化は，提案の空間分割シアーワープ法（Spatially Partitioned Shear-Warp Method）法により達成する．

1. まえがき

科学技術上の諸問題を計算機を用いて解き，結果を直感的に把握したいとの要求，あるいは高度医療における MRI，CT, 超音波診断装置などのデータの表示，などから大容量の３次元情報を可視化する高速かつ安価なボリュームレンダリングシステムが必要とされている．ボリュームの可視化は，ボリュームが物質の光透過率などの情報を３次元空間内の格子点上に持つため大量のデータ処理を必要とし， 高速表示には高価なスーパコンピュータや専用ハードウエアが用いられている．本論文は，市販の PC を複数用いて構成する低価格かつ処理性能がスケーラブルな並列ボリュームレンダリングシステムを提案する．このシステムの高速化は，提案の空間分割シアーワープ法（Spatially Partitioned Shear-Warp Method）により達成する．

2. 並列ボリュームレンダリング技術の現状

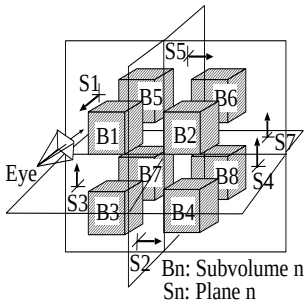
対話的な速度でボリュームレンダリングを行う方法として，専用ハードウエア（H/W) による方法 [5)6)8)10)~12)]，複数の演算装置を用いたソフトウエア (S/W) による方法 [9)14)15)] などが提案されている．専用 H/W によるレンダリングの実現は，高速処理可能であるが処理の柔軟性に欠ける． たとえば，PC クラスで初めて商用化を実現した VolumePro500[12)] では，平行投影のみが対象であり，また最適化の一つであるアーリーレイターミネーション (Early ray termination) が実現されていない．

いっぽう S/W による実現は，処理の自由度は高いが速度が遅い．高速化を図るには並列処理が用いられる．この方法には Binary-swap compositing 法 [14)] および The UltraVis System[15)] などがある．Binary-swap compositing 法は，ボリュームをサブボリュームに空間分割し，サブボリューム単位に映像発生および重畳するところに特徴があり，CM-5 などスーパーコンピュータを用いた並列化に関する提案である．しかし分割，すなわち並列数が２のべき乗であり，また投影法は平行投影に限られる．

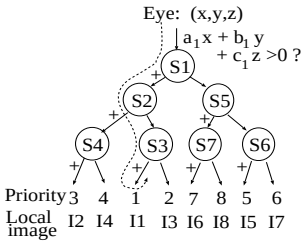
The UltraVis System[15)] は Pentium III による multi-processor platforms を対象にした方式で，キャッシュメモリーの積極的活用および MMX 命令等の命令レベルでの最適化を特徴とする．しかし，高速化のための最適化や各種チューニングが Pentium III 固有であり処理の変更が容易でない．

3. 提案する並列化の原理

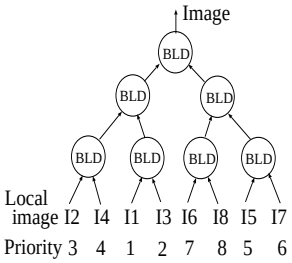
筆者らの提案する「空間分割シアーワープ法」（Spatially Partitioned Shear-Warp Method）は S/W による実現で



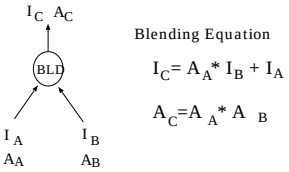
(a) ボリュームをサブボリュームに分割



(b) BSP によるサブボリュームの優先順位付番

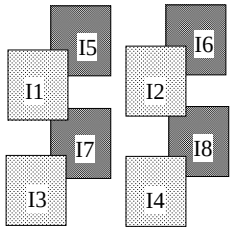


(a) 重畳木

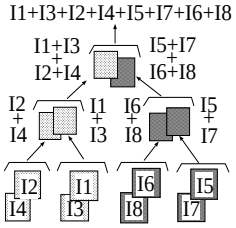


(b) ノードにおける重畳計算

図 1 BSP を用いた隠顕関係を示す優先順位の計算



(c) サブボリュームに対応した局所ベースプレーン映像



(d) 重畳木を用いたベースプレーン映像の合成

図 2 重畳木を用いた局所ベースプレーン映像の合成

あり空間分割による並列処理である点は，前述の Binary-swap compositing 法と同じある．しかし，提案法は，発生する局所映像がシアーワープ法 [9] におけるベースプレーン映像である点，重畳に 2 進木を用いるなどの点が異なる．このため，空間分割が偶数で良い，投影法に制限がない，といった利点を持つ．

空間分割シアーワープ法は，ボリュームを空間分割することによるシアーワープ法の並列化である．提案法は，シアー（Shear）したサブボリュームに対応して，各々のベースプレーン映像（歪んだ映像）を発生後，この映像間で隠顕関係を考慮して重畳（Composite）を行い，さらにこの重畳したベースプレーン映像にワーピング（Warping）を行い，スクリーン上に歪み補正された正しい映像を得る．このとき，ベースプレーン映像の発生および重畳が並列化される．

3.1 空間分割に伴う隠顕処理

サブボリューム単位に分割して映像発生を行う場合，隠顕関係（優先度）を考慮して重畳する必要がある．この重畳における優先順位の判断には，空間を分割し視点に近い順に優先度を付する方法である Binary space partitioning (BSP) 法 [1] をサブボリュームに適用する．全体をサブボリュームに分割して処理するため，計算ノード内のメモリー容量を一定とすれば計算ノード数の増加に伴ない，従来困難であった大規模ボリュームデータの表示が可能となる．

図 1(a) に視点と 8 分割したサブボリュームの優先順位を示す．このとき，分割数は必ずしも 2^n である必要はなく偶数であればよい．この優先順位を計算するのが図 1(b) に示した BSP である．すべてのベースプレーン映像（サブボリュームに対応する）に優先順位を付番するために，重畳木の根から分岐を繰り返してすべての葉をたどる．このとき，正側の枝への分岐か負側の枝かの判定は，各セパレート面を規定する (a_n, b_n, c_n, d_n) と視点位置 (x_e, y_e, z_e) を用いて計算した $a_n x_e + b_n y_e + c_n z_e + d_n$ の符号を用いる．(a_n, b_n, c_n) はセパレート面の法線，d_n は原点から面までの距離を示す．この計算結果が正の場合は，正側の枝に，負であれば負側の枝に分岐することを再帰的に繰り返し，すべての葉に優先順位を付番する．図 1(b) は，図 1(a) に示す位置に視点があるとした優先番号の付番結果を示す．図中の I_n はサブボリューム B_n から作られたベースプレーン映像を示す．

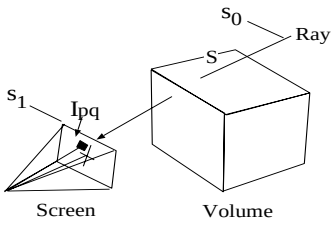
図 2(a) は，ベースプレーン映像を重畳する重畳木 (Compositing tree) を示す．図 2(b) は，重畳の式を示している．A 側の色 I_A と B 側の色 I_B を A 側の透明度 A_A により重畳している．この場合A側の画像がB側の画像の手前（すなわち優先度が高い）を前提としている．図 2(c) および 2(d) は重畳木の階層毎にどのように重畳が行われるかの模式図を示す．

3.2 光線の分割と合成

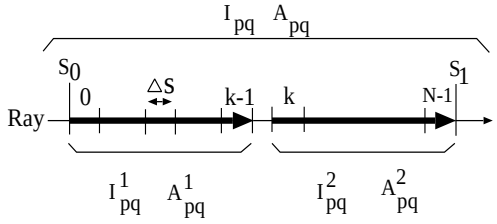
サブボリューム分割に伴ない分割された光線の合成に関して述べる．**図 3**(a) に，光線がボリュームを通過し，スクリーン上の該当画素位置 (p, q) に入射される様子を示す．このとき，スクリーン 上の画素 (p, q) 位置における輝度 I_{pq} は，

$$I_{pq} = \int_{s_0}^{s_1} L(s) e^{-\int_0^s \mu(t)\,\mathrm{dt}}\,\mathrm{ds} \qquad (1)$$

により与えられる [4]．ここに，$\mu(t)$ は光線上の位置 t における減衰率，$L(s)$ は位置 s における発光を示す．また，s_0 は光線の開始位置（パラメータ形式），s_1 は終了位置をそれぞれ示す．(1) 式における積分間隔を $\triangle s = \triangle t$ として離



(a) ボリュームと光線の関係



(b) 区間 $k-1$ および k における光線分割

図 3 並列処理のための光線分割

散化し，図 3(b) が示すように，光線を区間 k と $k-1$ の間で 2 分割したとすれば，

$$I_{pq} \quad = \quad I^1_{pq} + I^2_{pq} \cdot A^1_{pq} \qquad (2)$$

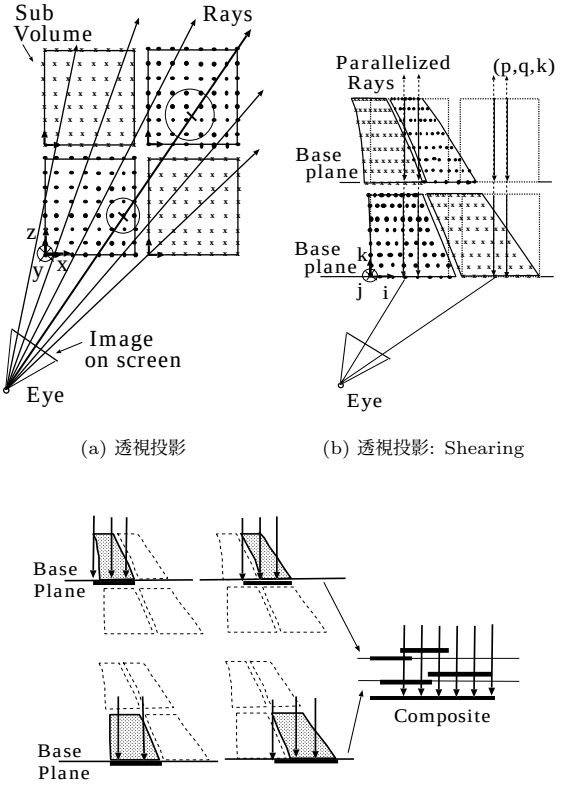
となる [11]．式中，I^1_{pq} は区間 0 から区間 $k-1$ を透過する光線の輝度，I^2_{pq} は区間 k から区間 $N-1$ を透過する光線の輝度，A^1_{pq} は区間 0 から区間 $k-1$ 間の累積透過率をそれぞれ示す．

これは，光線の減衰計算をサブボリュームごと独立に行い，それぞれの最終結果 I^1_{pq} および I^2_{pq} を累積透過率を用いて重畳すれば良いことを示す． 上式は，2 分割であるが任意の分割数に対しても成立する．

3.3 空間分割シアーワープ法

（1） 並列化

図 4 に提案する空間分割シアーワープ法の原理を示す．図 4(a) は透視投影の模式図を示す．透視投影の場合，アンチエイリアスの観点からサンプル点が視線から遠くなるに従い，サンプル値の決定に広い範囲のボクセル情報を用いる必要がある．図 4(b) は，光線を常時，基軸 (x,y,z 軸のうち視線に最も平行に近い軸）に平行とした場合の光線に対するボクセルの相対位置を示す．この性質を用いて，レンダリングする方法がシアーワープ法 [9] である． 筆者らの提案は，図 4(c) に示すように，各サブボリュームを視点に応じてシアー*し，各々の 2 次元ベースプレーン映像を発生後，重畳木を用いてベースプレーン間の重畳を行い 最終ベースプレーンを作り，この最終ベースプレーンを OpenGL のテクスチャマッピング機能を用いて歪み補正（ワーピング: warping）を行い，スクリーン上に正しい映像を得る．



(a) 透視投影　(b) 透視投影: Shearing

(c) 局所ベースプレーン映像の重畳

図 4 空間分割シアーワープ法： サブボリュームに対応する 2 次元ベースプレーン映像間で透明度を用いた重畳を行う

（2） 重畳処理位置

重畳処理をワーピング前のベースプレーン映像に対して行うか，ワーピング後のスクリーン映像に対して行うかは，システム構成上重要な選択である．提案法は，ベースプレーン映像間での重畳処理を採用した．これは，ワーピング処理の高速化を図るためグラフィックスボード内にハードウエアとして実装されているテクスチャマッピング機能を用いることを前提として，システムを設計したためである．このため，高価なグラフィックスボードはサーバーに 1 枚でよい．

（3） 分割境界での表示不良対策

対象点における輝度計算に用いる法線は周辺のボクセル値を用いて計算する．このため，ボリュームをそのまま分割したのでは境界部における法線が隣接するサブボリューム間で不連続になり，各サブボリュームに対応する部分映像の境界部における輝度に不連続性を生じる．この問題を

* リサンプル時にサンプル点とシアー後のボクセル位置の距離に応じた重みを求めるのに用いるのみであり，実際にボクセルの空間位置を変化させるものではない．

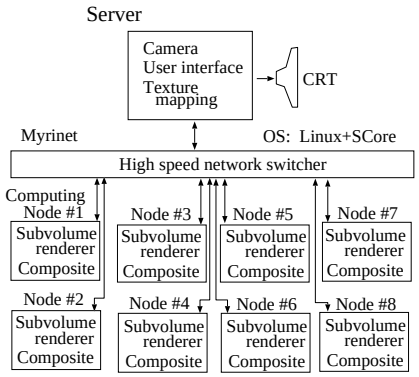


図 5 並列ボリュームレンダリングシステムの構成

表 1 構成した PC クラスタおよび要素 PC の性能諸元

No.	Block	Specification
(1)	CPU	Pentium III 700 MHz (8PC x 2CPU)
(2)	Memory	512MB
(3)	Network	Myrinet 1Gbits/sec
(4)	OS	SCore 新情報処理開発機構 Linux kernel 2.2.14 with XF86 3.3.6
(5)	Graphics API	OpenGL and GtkGLarea

防ぐため，各ノードの持つサブボリュームは，実分割サイズより各軸方向に 1 ボクセル大きなボリュームとしている．この 1 ボクセルは隣接するサブボクセルとの重複部であり，法線の計算に用いるが，発生する部分映像のサイズには影響を与えない．

4. PC クラスタによる実装

4.1 システム構成

提案法を PC クラスタ上に，MPI(Message Passing Interface) を用いた並列処理により実装した．**図 5** に PC クラスタの構成と各ノードの受け持つ処理を示す．各 PC はサブボリューム単位のレンダリングおよび重畳処理を行う．各 PC 間の通信は MPI を用いた．物理的通信の高速化を図るため，PC 間接続は Myrinet（転送速度 1G ビット/秒) を用いている．**図 6** に並列ボリュームレンダリングシステムとして構成した PC クラスタの概観を示す．また，構成した PC クラスタ（16 ノード）および要素 PC の性能諸元を**表 1** に示す．実装したプログラムは PC 数の増加に対して*コードの変更なく実行可能である．

4.2 処理概要

空間分割シアーワープ法におけるサーバおよびノードの処理フローを**図 10** 示す．

5. 性 能 評 価

本提案法および Ray-casting 法 [7] を PC クラスタに実装

* 評価 PC クラスタの並列度は 1 6 ノードであるが，16 ノード以上にもコードの変更なく処理可能である．

図 6 PC クラスタを用いて実装したレンダリングシステムの概観

し，ボリュームデータサイズ，表示画像サイズおよびノード数をパラメータとして，レンダリング時間（重畳を含む）および重畳時間を計測した． 比較に用いるため実装した並列 Ray-casting 法は，**図 11** に示すように，サブボリューム単位に Ray-casting 法を並列化して局所映像（スクリーン上の部分映像に対応）を発生し，これを重畳木を用いて合成する．すなわち，部分映像の発生が Ray-casting 法であり部分映像の重畳処理は提案法と同じとして実装した．また，Ray-casting 法は直接スクリーン上の部分映像が得られるためスクリーン上の映像間で重畳を行っている．

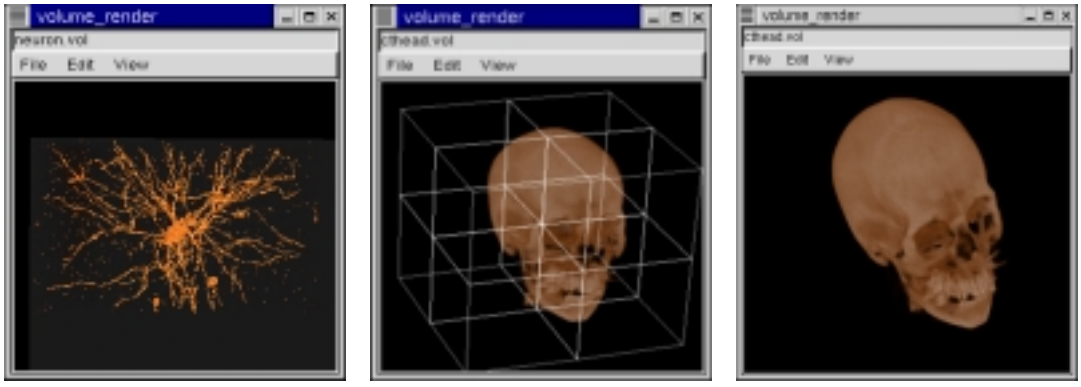
表 2 に評価に用いたボリュームデータの諸元および評価条件を示す．また，**図 7** に評価対象データの提案法による出力映像を示す．評価として，**表 2** に示す各条件において並列度を変化させ両方式のレンダリング時間および重畳処理時間を比較した．**図 8**(a) に実験条件 (1) における両方式のレンダリング時間比較を示す．また，図 8(b) に実験条件 (2 ）における両方式のレンダリング時間比較を示す．いずれの場合もノード数を 2，4，8，16 と変化させてレンダリング時間（重畳時間を含む）を比較した．本提案法は Ray-casting 法に比較してレンダリング速度が速いとの実験結果が得られた．

図 9(a) に実験条件 (1) における両方式の部分映像重畳処理に要した時間の比較を示す．また，図 9(b) に実験条件（2 ）における両方式の同様な比較を示す．いずれの場合もノード数を 2，4，8，16 と変化させて重畳時間を比較した．

5.1 実験の考察

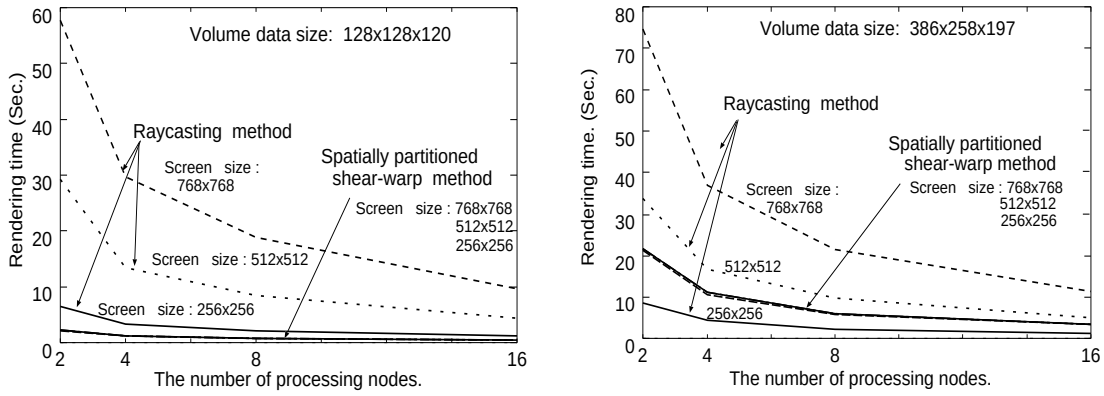
（1） 画 質

図 7 に示す提案法による表示映像は何れも 8 並列での処理による出力を示す．図 7(b) には，8 並列時の部分空間分割を線画により示す． 空間分割したサブボリュームを各々



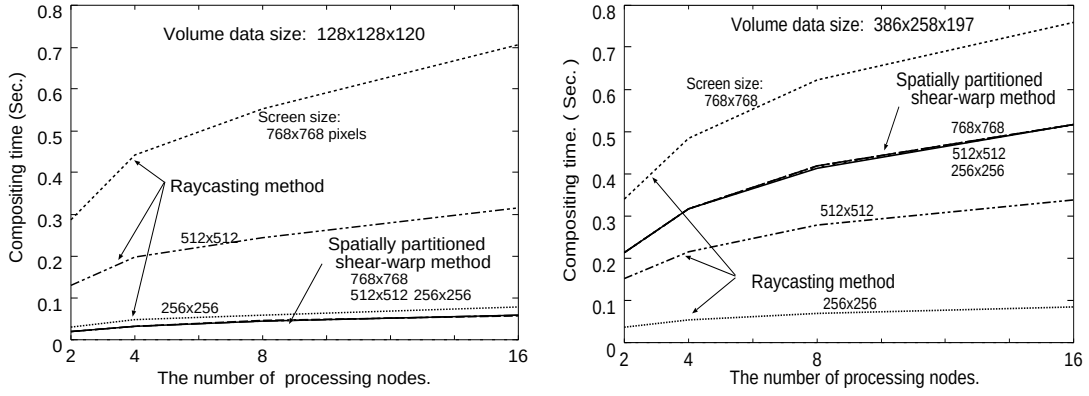
(a) ニューロン　(b) 人体頭部（8サブボリューム分割による並列処理を示す）　(c) 人体頭部

図 7 空間分割シアーワープ法による評価ボリュームデータの表示映像 （8 並列）



(a) 頭部データを用いた処理時間比較　(b) ニューロンデータを用いた処理時間比較

図 8 提案法および Ray-casting 法の並列度による処理時間比較



(a) 頭部データを用いた重畳時間比較　(b) ニューロンデータを用いた重畳時間の比較

図 9 提案法と Ray-casting 法の並列度による重畳時間比較

表 2 時間計測の組み合わせ条件

実験条件番号	ボリュームデータ	表示画素数
(1)	人体頭部 (128x128x120 ボクセル)	256^2
		512^2
		768^2
(2)	ニューロン (386x258x197 ボクセル)	256^2
		512^2
		768^2

```
//================== サーバの処理 ========================//
ProcessServer()
{
  SendVolumeData() ;          // ノードにサブボリュームを送る.
  // フレームの繰り返し.
  for(;;){
    BroadcastCameraEtc();   // 視点情報などをノードに送る.
    ReceiveBasePlane();     // 最終合成ベースプレーンの受け取り
    //最終合成ベースプレーンをテクスヤマッピングのプレーンとして
    // ベースプレーンの内容をテクスチャパターンとしてテクスチャ
    //  マッピングする.( OpenGL を利用)
    TextureMapping();
  } // 無限ループ
}

//================== ノードの処理 ======================//
ProcessNode()
{
    // サブボリュームデータを読む.
  ReadSubvolume()

    // フレームの繰り返し.
  for( ;;) {
      // サーバからカメラ情報などの受信を待つ.
    WaitCameraEtc();
      // ベースプレーン (BP) の初期化
    ClearBasePlane( ) ;
      // サブボリュームに割り当てられたボクセルを i,j,k 順にすべてたどる.
      //   (i,j,k) は BP に直角な基線をｋ軸, BP の辺を i,j 軸とする右手座標系.
      //   変数 begin?, last?がサブボリュームの範囲を示す.
    for ( k = beginK ; k  <=  lastK ; k ++ ) {
        // 変数 update に更新に必要な起点および変化量などの情報を持つ.
        // (p,q) は点 (i,j,k) がシアーされた位置 (BP 上のメモリー位置)
      update  = GetUpdateParam( k ) ;
        // スライスｋにおける左上ボクセル位置がシアーを受けて移動
        //   した点の J 座標を起点ｑとして加算によりアップデートする.
      q = update.beginShearedJ  ;
      for (  j= beginJ   ; j <= lastJ   ; j++ ) {
          //  J方向のシアー位置を変化量の加算により更新する.
        q += update.deltaJ ;
        p = update.beginShearedI ;      // I 方向の起点位置の設定
        for (  i = beginI ; i <= lastI ; i++ ) {
          p += update.deltaI ;          // 変化量により更新
            // ijk はボクセル位置であり, sijk は点 ijk がシアーされた位置
          ijk =( i,j,k)  ;      sijk=( p,q,k) ;
            // ボクセル位置 ijk を用いてボクセル情報を取り出し, 書き込み
            //   位置 (整数化 p, 整数化 q) と (p,q) の距離を用いて書き込み情報
            //   の寄与率を考慮して pixel を求める. pixel は r,g,b,a で構成.
          pixel = shear.ResampleWithConvolution( ijk , sijk ) ;
            //   pixel(r,g,b,opacity) を (p,q) 位置で決まるベース
            //   プレーンにアルファブレンドで書き込む. メモリアドレスの計算.
          pmem = (int ( q  ) * sizeijk.bpwidth  + int(p))*4 ;
          CompositeBasePlane(  pmem, pixel ) ;
        }  // i ループの終了
      }  // j ループの終了
    }  // k ループの終了

    // 重畳ポイントがツリーの先頭になるまで以下を実行する.
   while ( notRoot ){
     RecieveBasePlane();     // BP を受信する.
     LocalMergeBasePlane() ;  // 受信 BP と自ノードが持つＢＰで重畳処理を行
う.
     SendBasePlane();        // 次の重畳ポイントに対応するノードへＢＰを送る.

   }
 } // フレームの繰り返し.
}
```

図 10 PC クラスタに実装した空間分割シアーワープ法の処理

レンダリングして部分映像を発生して合成するため境界付近における映像の不連続性が心配になるが，図 7(a) および (c) からわかるように境界部分付近での映像表示に不連続性は見られない．

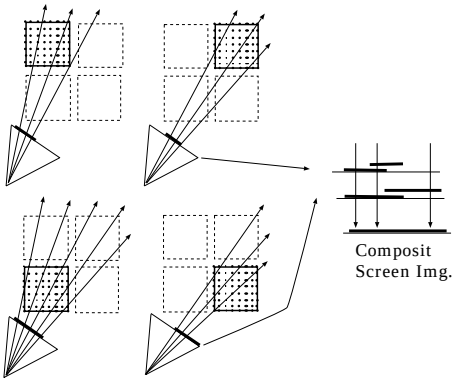


図 11 レンダリング時間比較のために PC クラスタに実装した並列 Ray-casting 法

（2） レンダリング速度

ボリュームの一辺のサイズを V としたとき，レンダリングに要する計算量のオーダは $O(V^3)$ である． いっぽう Ray-casting 法はスクリーン上の画素からレイを出すためスクリーンサイズを P ピクセルとしたとき計算量のオーダは $O(P^3)$ [*] である．このため，シアーワープ法の処理時間はボリュームデータのサイズにより，Ray-casting 法はスクリーンサイズに依存する．この傾向は測定結果，図 8 に現れている． 本提案法ではサブボリューム分割を均等に行うので，計算ノード数を N_p としたとき，単純に対象サブボリュームサイズが $\frac{V^3}{N_p}$ となり処理の線形性を持つ．

両方式の表示速度比は P^3/V^3 であり，図 8(a) と (b) との比較によれば V の増加が提案法の性能低下を招いているように思えるが，次のような理由から問題はないと考える．通常の映像表示（好ましい状態での）では, スクリーンサイズ P およびボリュームサイズ V は独立ではなく，P/V が一定と考えられる．即ち，表示分解能が高い場合は，用いるボリュームデータの分解能も高いはずである．従って，この状態では P/V は一定と考えられためボリュームサイズ V の増減に関係なく，提案アルゴリズムの処理は常に Ray-casting に比べ高速である．図 8(a) と (b) の単純比較では，スクリーンサイズ P およびボリュームサイズ V を独立に比較するため，ボリュームが大きくなると Ray-casting との差が小さくなる．

（3） 部分映像の重畳時間

部分映像の重畳時間に関しては図 9 に示すように計算ノードの増加に伴ない増加している．これは次のように解釈できる．

[*] 奥行き方向のサンプリング間隔を V と考えれば $O(P^2V)$ と考えられるが，議論を明確にするため奥行き方向のサンプリング間隔を S 程度と仮定し計算量を $O(P^3)$ とした．

重畳処理を，混合計算（2）式の実行時間 T_m（2 枚のサブボリュームイメージを重畳する時間）と他ノードへの転送時間 T_t（1 枚のサブボリュームイメージを転送する時間）に分けるとすれば，重畳に要する総時間の下限 T は，重畳および通信の回数 N_c を用いた次式になる．

$$T \quad \leqq \quad N_c(T_m + T_t) \tag{3}$$

係数 N_c は重畳木の葉の数を N_p としたとき重畳および通信回数の下限であり，

$$N_c = N_p(2^{-1} + 2^{-2} + \ldots + 2^{-log_2 N_p}) = N_p - 1$$

である．

転送時間 T_t はデータの転送時間 T_x と転送セットアップ時間 T_{set}（転送時間以外の時間）に分解されるので, 上の式は

$$T \quad \leqq \quad N_c(T_m + T_{set} + T_x) \tag{4}$$

これを N_p ノード数で並列処理するとすれば，T_{set} は並列化できないので並列化した重畳時間の下限 T_{N_p} は，

$$T_{N_p} \quad \leqq \quad (N_p - 1)(\frac{T_m + T_x}{N_p} + T_{set}) \tag{5}$$

N_p を大きくとれば，

$$T_{N_p} \quad < \quad T_m + T_x \quad + (N_p - 1)T_{set} \tag{6}$$

となる．この式は，図 9(a)，(b) の実験結果を近似する．

6. むすび

本論文は，対象ボリュームを部分空間であるサブボリュームに分け，部分空間単位にシアー法を用いて部分映像を発生し，これを 2 進木で合成したあとワーピングする「空間分割シアーワープ法」を提案する．PC クラスタ上に実装した提案法および Ray-casting 法との性能比較によれば，提案法は高速化を達成する．スクリーンサイズ 768^2 ピクセルおよびボリュームデータを 256^3 としたとき，約 4 倍程度高速である．また処理性能のスケーラビリティも持つ．

提案法は，並列度に応じて全体ボリュームをサブボリュームに分け，各ノードに分配して処理するため，計算ノード内のメモリー容量を一定とすれば，計算ノード数を増加することにより，従来困難であった大規模ボリュームデータの可視化が可能となる．

いっぽう処理のスケーラビリティに関しては計測した範囲では良好であるが，高並列に対する解析は十分でない．今後，既存の大規模 PC クラスタを用いて 512 並列に対する処理時間の計測および評価を行う予定である．

謝　辞

本研究で技術支援を頂いた，技術研究組合 新情報処理開発機構 並列分散システムソフトウエア つくば研究室長 石川 裕 氏に深謝する．

〔文　献〕

1）H. Fuchs, Z, Kedem, and B,Naylor. On Visible Surface Generation by A Priori Tree Structures. In Proceedings of the ACM SIGGRAPH ’80 Conference, pp.124-133 (July 1980)
2）S. Muraki, K. Shimokawa, M. Ogata, K. Kajihara, K. Ma and Y. Ishikawa: “VG Cluster: Large Scale Visual Computing System for Volumetric Simulations, SC2000 Research Gems(Poster), November 2000
3）S. Muraki, M. Ogata, K. Ma, K. Koshizuka, K. Kajihara, X. Liu, Y. Nagano and K. Shimokawa: “Next-Generation Visual Supercomputing using PC Clusters with Volume Graphics Hardware Devices,” will be appear on SC2001: Super Computer Conference, Dever, Colorad, November 2001
4）A. S. Glassner: “Principles of Digital Image Synthesis”, Morgan& Kaufman (1995)
5）T. Günther, C. Poliwoda, C. Reinhart, J. Hesser, R. Männer, H.-P. Meinzer, and H.-J. Baur: “Virim: A massively parallel processor for real-time volume visualization in medicine”, Computers & Graphics, 19, 5, pp.705-710 (1995)
6）A. Kaufman and R. Bakalash: “Memory and processing architecture for 3D voxel-based imagery”, IEEE Computer Graphics & Applications, 8, 6, pp. 10-23 (Nov. 1988
7）M. Levoy: “Efficient ray tracing of volume data”, ACM Transactions on Graphics, 9, 3, pp. 245-261 (July 1990)
8）G. Knittel and W. Strasser: “A compact volume rendering accelerator”, In Proceedings of the IEEE Symposium on Volume Visualization, pp. 67-74 (Oct. 1994)
9）P. G. Lacroute and M. Levoy: “Fast volume rendering using a shear-warp factorization of the viewing transformation”, In Proceedings of the ACM SIGGRAPH ’94 Conference, pp. 451-457 (July 1994)
10）M. Ogata, H. Ohkami, H.C. Lauer and H. Pfister: “A Real-Time Volume Rendering Architecture with Resampling Scheme for Parallel and Perspective Projections”, Proceedings of the ACM/IEEE Symposium on Volume Visualization, pp. 20-29 (Oct. 1998)
11）緒方, 梶原, 村木, 下川, マ, 土肥, “マルチプルシア重畳法を用いた並列ボリュームレンダリングシステム”, 映像情報メディア学会誌 Vol.55 No. 7, pp.1011-1018, 2001 年 7 月
12）H. Pfister, J. H. Hardenburg, H. Lauer and S. Sailor: “The VolumePro Real-Time Ray-Casting System”, In Proceedings of the ACM SIGGRAPH ’99 Conference, pp. 131-138 (Aug. 1999)
13）L. Williams: “Pyramidal parametrics”, In Proceedings of the ACM SIGGRAPH ’83 Conference, pp. 1-11 (July 1983)
14）Kwan-Liu Ma: “Parallel Volume Rendering Using Binary-Swap Compositing”, In IEEE Computer Graphics and Applications , 14, 4, pp. 59-68 (July 1994)
15）G. Knittel: “The UltraVis System”, In Hewlett-Packard Laboratories, Usual Computing Department, knittel@hpl.hp.com