JAERI-Data/Code
2000-021

実時間可視化システムのための ボリュームレンダリング・モジュールの開発

2000年3月

大谷孝之・村松一弘

日本原子力研究所
Japan Atomic Energy Research Institute

本レポートは、日本原子力研究所が不定期に公刊している研究報告書です。
入手の問合わせは、日本原子力研究所研究情報部研究情報課（〒319-1195　茨城県那珂郡東海村）あて、お申し越し下さい。なお、このほかに財団法人原子力弘済会資料センター（〒319-1195　茨城県那珂郡東海村日本原子力研究所内）で複写による実費頒布を行っております。

This report is issued irregularly.
Inquiries about availability of the reports should be addressed to Research Information Division, Department of Intellectual Resources, Japan Atomic Energy Research Institute, Tokai-mura, Naka-gun, Ibaraki-ken 〒319-1195, Japan.

編集兼発行　日本原子力研究所

JAERI－Data/Code 2000－021

実時間可視化システムのためのボリュームレンダリング・モジュールの開発

日本原子力研究所計算科学技術推進センター
大谷　孝之・村松　一弘

（2000年2月9日受理）

ボリュームレンダリングとは、３次元の解析空間における物理量の分布について任意の視点から光線追跡を行い、物理量の境界面のみならず内部の情報構造も透かしてみることの出来る可視化手法である。

そのため、科学技術計算における計算結果データの解析等の際には、非常に有用な可視化機能であるが、反面、他の可視化に比べて計算時間が膨大になるという短所を持つ。

本報告では、並列計算機上での流体解析のための実時間可視化システムの機能の一つとして開発したボリュームレンダリング・モジュールについて記述する。このモジュールは構造格子を用いて計算された結果を直接可視化し、発見的な手法を用いることでレンダリングの一部を高速化している。さらに、並列処理によるボリュームレンダリングの高速化についても考察する。

日本原子力研究所：〒153-0061　東京都目黒区中目黒2-2-54

JAERI－Data/Code 2000－021

Development of Volume Rendering Module for Real-Time Visualization System

Takayuki OTANI and Kazuhiro MURAMATSU

Center for Promotion of Computational Science and Engineering
Japan Atomic Energy Research Institute
Nakameguro, Meguro-ku, Tokyo

(Received February 9, 2000)

Volume rendering is a method to visualize the distribution of physical quantities in the three dimensional space from any viewpoint by tracing the ray direction on the ordinary two dimensional monitoring display. It enables to provide the interior information as well as the surfacial one by producing the translucent images. Therefore, it is regarded as a very useful means as well as an important one in the analysis of the computational results of the scientific calculations, although it has, unfortunately, disadvantage to need a large amount of computing time.

This report describes algorithm and its performance of the volume rendering software which was developed as an important functional module in the real-time visualization system PATRAS. This module can directly visualize the computed results on BFC grid. Moreover, it has already realized the speed-up in some parts of the software by the use of a newly developed heuristic technique. This report includes the investigation on the speed-up of the software by parallel processing.

Keywords: Scientific Visualization, Computer Graphics, 3D Image, Ray Casting, Simulation

目　　次

Contents

1. はじめに

近年、計算機の高性能化に伴い科学技術分野における数値シミュレーションの大規模化・高詳細化が顕著になってきた。このため、科学技術計算の結果生じるデータは膨大となり、直接数値データを人間が読んで解釈することは不可能になってきた。したがって、計算結果の解析にはデータの可視化が不可欠となっている。

こうした需要に伴い計算結果の可視化ツールが開発、市販されるようになったが、計算結果の可視化については、解析計算が全て終了した後に解析データを可視化するのみならず、デバッグやデモンストレーション等の目的で計算結果を計算実行中に（理想的には実時間で）モニタリングする要求や、さらには解析計算実行の途中で計算パラメータを変更したいという要求が生じてくる。

日本原子力研究所（以下、原研）では、こうした要求に応えるべく、実時間可視化システムPATRAS[1, 2]を開発している[1]。このシステムは、任意の並列計算機上で実行可能なように計算機間のポータビリティを考慮して、言語としてFORTRAN及びC、プロセッサ間通信として標準的なMPI（Massage Passing Interface）を使用している。また、計算の実行や可視化は、実際に解析計算を行う計算機（計算サーバ）とネットワークで接続された端末（パソコンやワークステーション等）から操作可能になっており、この際端末の機種・OSに依存しないようJavaアプレットを用いて、Webブラウザさえインストールされていれば利用可能になっている。

このシステムでは、可視化機能として、オブジェクト表示、等高線表示、ベクトル図表示、等値面表示、パーティクルトレースの機能を備えている。

一方、現実の時空間や自然現象、計算モデルにおけるオブジェクトや計算結果の多くは、3次元のボリュームデータとして表現される。ボリュームデータは、内部の情報が詰まった3次元の立体（喩えて言うなら多種の果汁を固めたゼリーのようなもの）であるので、ポリゴンや線分といった幾何要素からは本質的に構成不可能な場合や幾何学的に表現するには情報量が多過ぎる場合が多い。このようなボリュームデータを表現（可視化）するには、可視化対象の形状を面や線の組み合わせとして表現するCG手法（サーフェイスレンダリング）では不十分となる。

また、3次元のボリュームデータの可視化においては、通常は見ることの出来ないボリュームデータ内部を透かして見せることにより、解析対象の時空間や自然現象の複雑な構造や振る舞いを理解するための視覚的な洞察手掛かりを提供することが重要である。

このため、本報告では前記の実時間可視化システムに、ボリュームデータ内の物理量に任意の透明度を定義することにより半透明でボリュームデータ内を透かしてみることの出来る可視化機能を付加すべくボリュームレンダリング・モジュールを開発した[3]。

ボリュームレンダリングは、半透明で内部の情報が含まれるため生成される画像1枚当た

[1]原研とNECとの共同研究により開発されている。

りの情報量が多いという特長を持つ反面、ボリュームの内部に至るまで光線の反射・吸収計算を行う必要があるため画像を生成するための計算時間及び必要メモリ量が大きいというデメリットを持つ。このデメリットを回避するため、なるべく高速にレンダリング処理を行うアルゴリズムについて検討した。また、さらなる高速化のために、ボリュームレンダリングを並列処理することを念頭に置き、その並列化法について定性的な検討を加えた。

2. 実時間可視化システムの概略

2.1 システムの目標

一般に、大規模計算プログラムでは、計算結果として大量の数値データが生成される。このため、このようなプログラムでは、計算結果あるいは経過を可視化しないと、計算に誤りがないかどうかのチェックや計算結果の解析を行うことが出来ない。

原研において現在開発中[2]の実時間可視化システムは、並列計算機上で計算された結果を高速に（実時間で）可視化し、プログラマの手元の端末に表示するとともに、プログラマの手元の端末から容易に計算パラメータを変更することを可能とするものである。

したがって、このシステムを利用すれば、既存の多くの可視化ソフトウェアのように計算が終了してから可視化するのではなく、計算実行中に順次計算結果が可視化されるうえに、その結果を視覚的に確認しながら計算途中で計算パラメータを変更することができるので、より効率的なプログラムの開発、デバッグ、計算結果の解析が可能となる。

また、このシステムは、手元の端末からネットワークを通じて並列計算機上での計算を実行、可視化を行うので、例えば学会の会場や出張先などの遠隔地からノートパソコンで計算を実行させ、その結果（画像）をノートパソコンのディスプレイに表示することができる。

さらに、この可視化システムは、並列計算機上のプログラムの中にサブルーチンをコールする要領で呼び出し文を書き加えることで計算結果を可視化するので、任意のユーザのプログラムへの組み込みも容易だと思われる。

2.2 システムの構成

Fig.2.1にシステムの全体構成（計算機が分散メモリ型並列計算機の場合の例）を示す。システムは、解析計算・画像生成計算を行う計算機（計算サーバ）、ユーザの手元の端末（webクライアント）、計算サーバとwebクライアントを中継するwebサーバからなっている。

計算サーバ内で計算されたデータは直ちに可視化され、生成された画像はwebサーバの画像データ中継モジュールを通じてwebクライアントのディスプレイに表示される。一方、webクライアント上で計算実行や計算パラメータの変更を行うとそれらはwebサーバのパラメータ中継モジュールを通じて計算サーバでの計算に反映される。

図示したシステムでは、並列計算機の特長を活かして、解析計算・画像生成を行うプロセッサと画像合成・圧縮を行うプロセッサとを分け、パイプライン（流れ作業）的に処理することによって画像生成の高速化を図っている。

次に、ピクセルデータ（画像データ）並列生成部の構成をFig.2.2に示す。ピクセルデータ生成部は解析計算プログラムからライブラリモジュールを呼び出すことにより実行され、図

[2]原研とNECとの共同研究により開発されている。

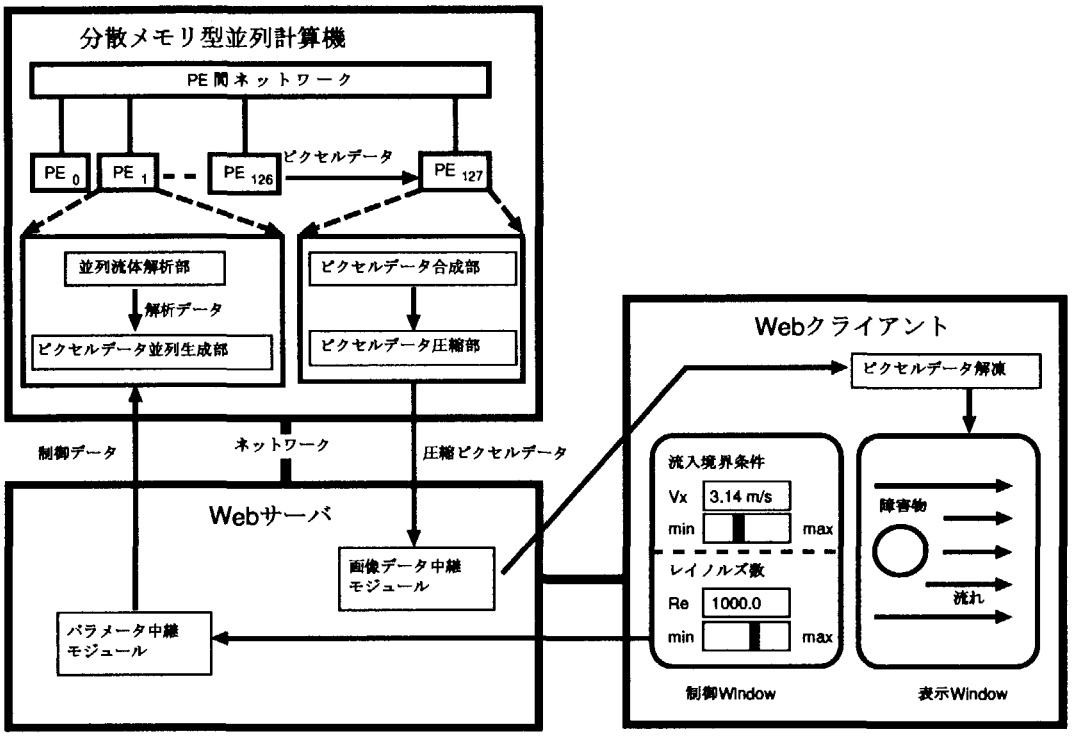


Fig. 2.1 System configuration of JAERI real-time visualization system.

種としてはオブジェクト表示（流れ場の障害物等の３次元物体の表示）、等高線図、ベクトル図、パーティクルトレースおよびそれらの重ね合わせが可能である。

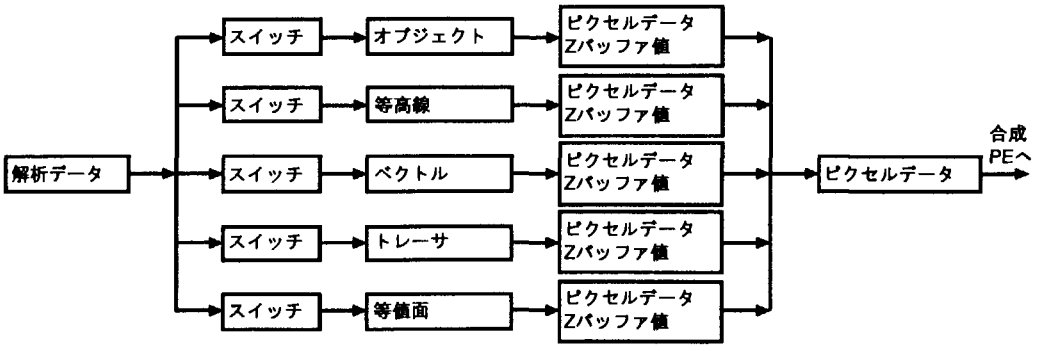


Fig. 2.2 Pixel data generation modules of JAERI real-time visualization system.

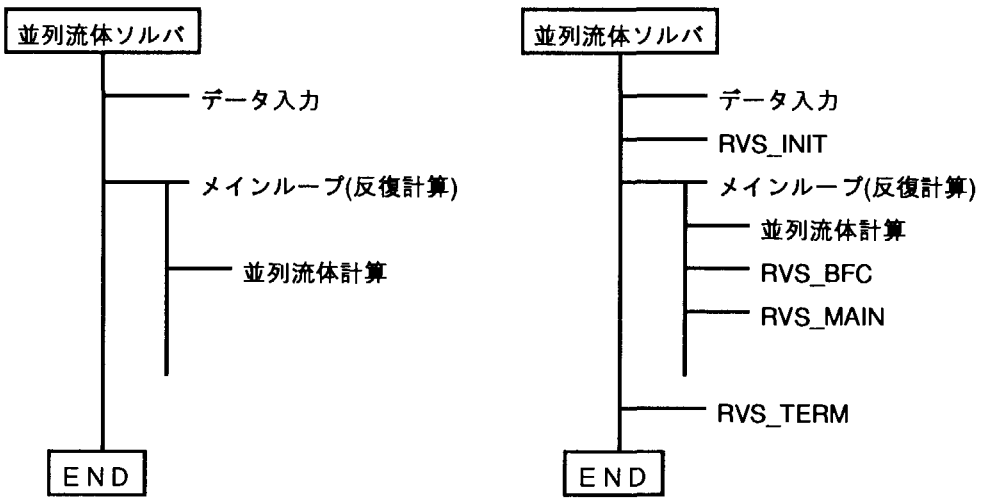
また、ピクセルデータ生成部は、ほぼすべて標準FORTRAN77およびCで記述されており、高いポータビリティ（異種計算機間のソフトウェア移植性）を備えている。

2.3 システムの操作

2.3.1 ライブラリの呼び出し

この可視化システムは、ライブラリモジュール化されており、ユーザの解析計算プログラムからのサブルーチンコールによって呼び出される。例えば、解析計算プログラムがBFC格子[3]を採用している場合、基本的に以下の４つのサブルーチンを呼び出すことによって可視化が実現される（Fig.2.3参照）。

RVS_INIT: 可視化機能の初期化及び画像生成用モジュールの起動
RVS_BFC : BFC格子における格子データ、解析データのRVS_MAINへの引き渡し
RVS_MAIN: 可視化処理及び表示の制御
RVS_TERM: 可視化機能の終了処理



(a) 可視化ルーチン呼び出し前　　(b) 可視化ルーチン呼び出し後

Fig. 2.3 Library routine call procedure.

可視化機能を組み込んだシステムの作成は、具体的にはユーザの解析計算プログラムにRVS_INIT、RVS_BFC、RVS_MAIN、RVS_TERMなどの呼び出し文を書き込み、さらにユーザ定義ファイルを作成して、これらのファイルをコンパイルし、可視化システムのオブジェクトファイルとリンクする。

2.3.2 ユーザインタフェース

このシステムでは、ユーザがより簡便に可視化操作を行えるようGUI（Graphical User Interface）を通して可視化を制御する。このシステムでは、webクライアント（端末）のOS

[3]Boundary-Fitted Curvilinear Grid：構造格子（格子点が規則正しく並んだ格子）の一種。格子線を計算領域の境界線と一致させた格子。

This is a blank page.

（Operating System）に依存しないGUIを実現するためにJavaアプレットを採用している。

したがって、ユーザの手元の端末には、Java対応のブラウザ（Netscape Navigator、Internet Explorerなど）さえあればよく、OSには無関係である。つまり、端末はWindowsのパソコンでもMACでも、UNIXワークステーションでもかまわない。

Fig.2.4にGUIによる流体計算の可視化の例（電車車両まわりの流れ）を示す。ブラウザ内に表示されるボタンをクリックすることにより、可視化の制御ができる。

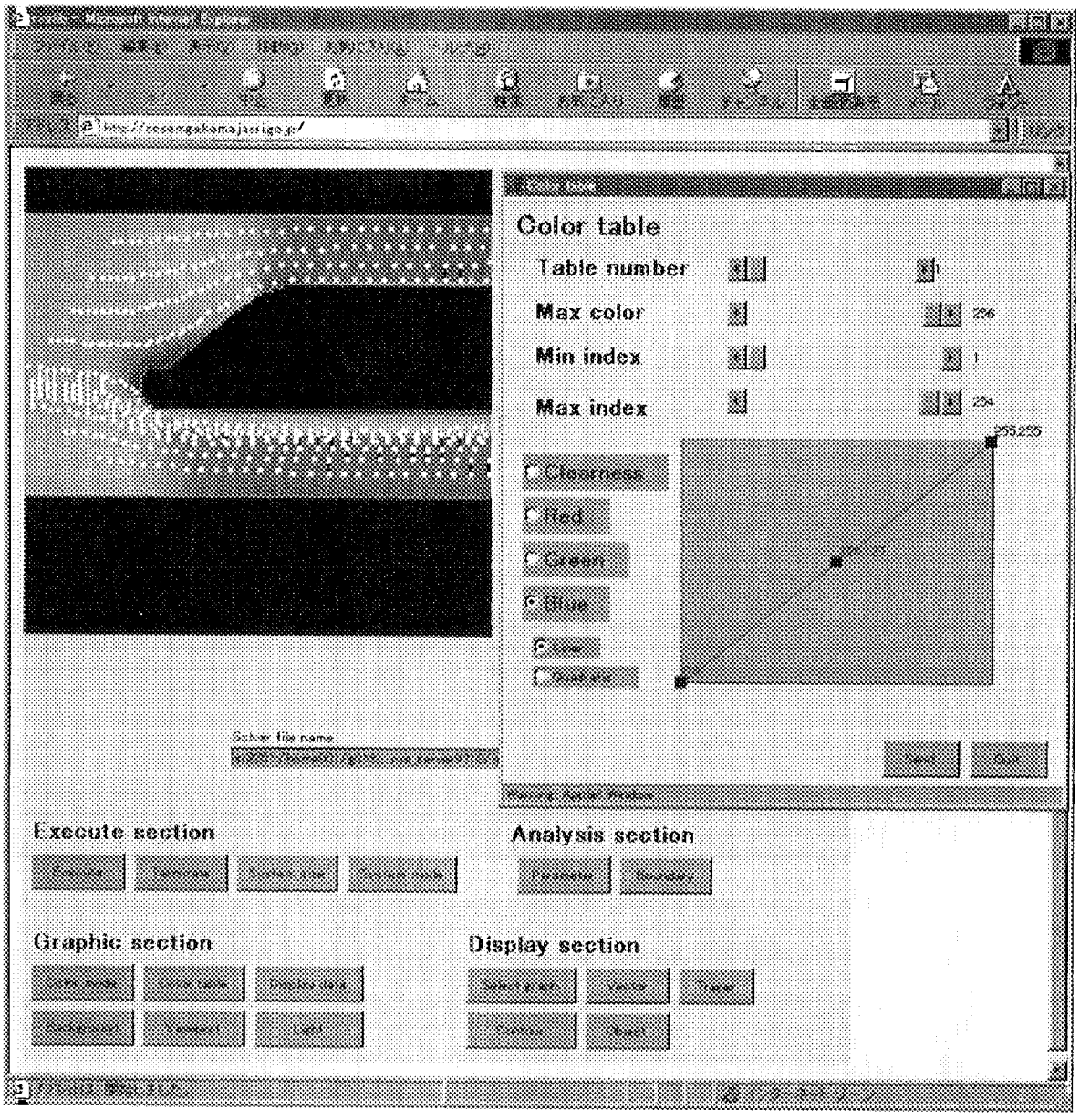


Fig. 2.4　GUI of JAERI real-time visualization system.

This is a blank page.

3. ボリュームレンダリングの原理とアルゴリズム

3.1 ボリュームレンダリングの原理

ボリュームレンダリングでは、計算にボクセルを使用する。ボクセルは可視化対象である３次元空間を構成する最小単位立体である。計算格子を用いる数値シミュレーションの計算結果を可視化したい場合は計算格子によって仕切られる単位立体がボクセルと見なされる。ボリュームレンダリングの原理を Fig.3.1 に示す [4, 5]。

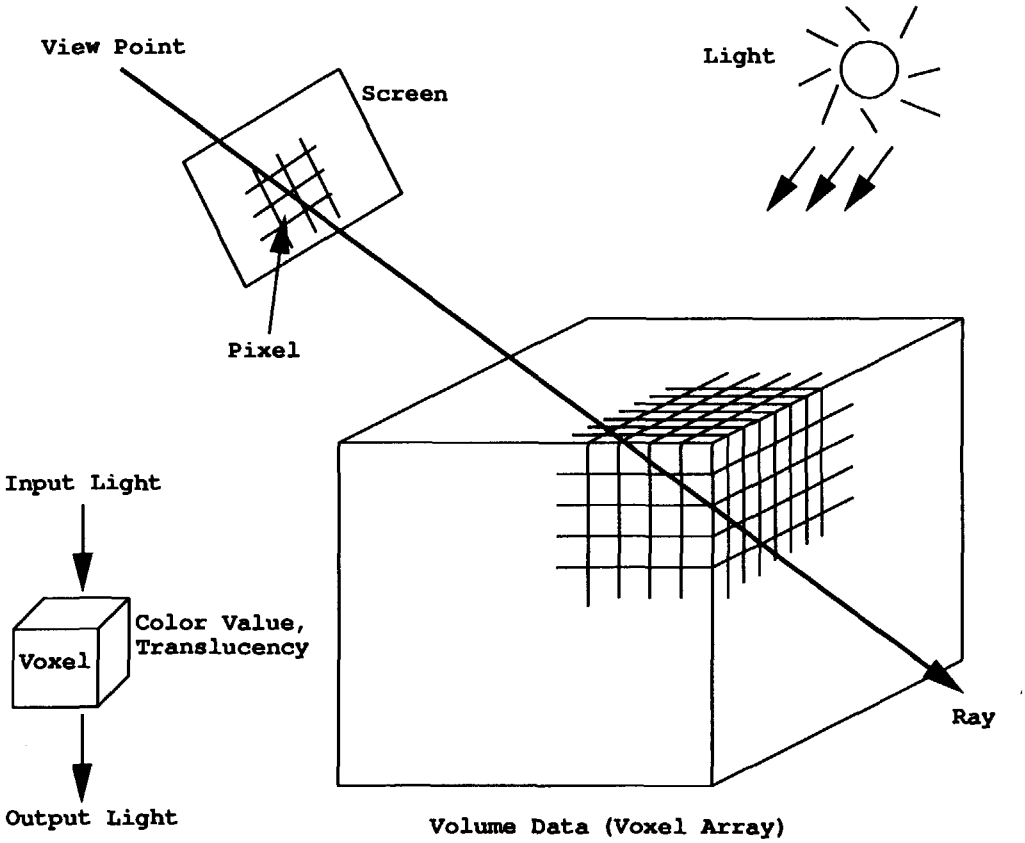


Fig. 3.1 Principle of volume rendering.

ボリュームレンダリングは、３次元空間（ボリュームデータ）内に分布しているボクセルをスクリーン上のある画素から発せられる視線にそってサンプリングし、その値（ボクセル中心における物理量に対して定義された色の値）を加算していくことで当該画素の色が決定され、全ての画素について同様の処理を行って最終的な画像を得る。ボリュームデータのような内部の詰まった立体に対してその内部構造の全体を把握するには、半透明表示が向いていることは直感的に自明である。ボリュームレンダリングで半透明表示を行う場合には、立

体を構成している各ボクセル（の物理量）に対し、透明度の要素を定義すればよい。

ボリュームレンダリングと一般のＣＧ手法（サーフェイスレンダリング）では、ボリュームデータ内のデータ構造やレンダリングに対する考え方が大きく異なる。一般のＣＧ手法では３次元空間に配置された面が基本になる。計算された３次元空間にたとえ内部構造があっても、物体表面とは別に内部に面が構成されていたにすぎない。これに対し、ボリュームレンダリングでは、計算している３次元物体の表面や内部といった考え方自体がない。あるのは３次元のボリュームだけで、どの部分が表面だとかどの部分が内部構造だとかといった区別はデータ上に存在しない。

また、通常のサーフェイスレンダリング（Fig.3.2）では、視点に最も近い表面を探し出し、その表面と視線との反射・吸収計算を行えば、それ以外の表面との計算は不要であるのに対し、ボリュームレンダリングでは視線と交差する全てのボクセルとの計算が必要であることに気付く。このため、ボリュームレンダリングでは、一般のＣＧ手法に比べて極めて多くの計算時間、記憶容量が必要となるのである。

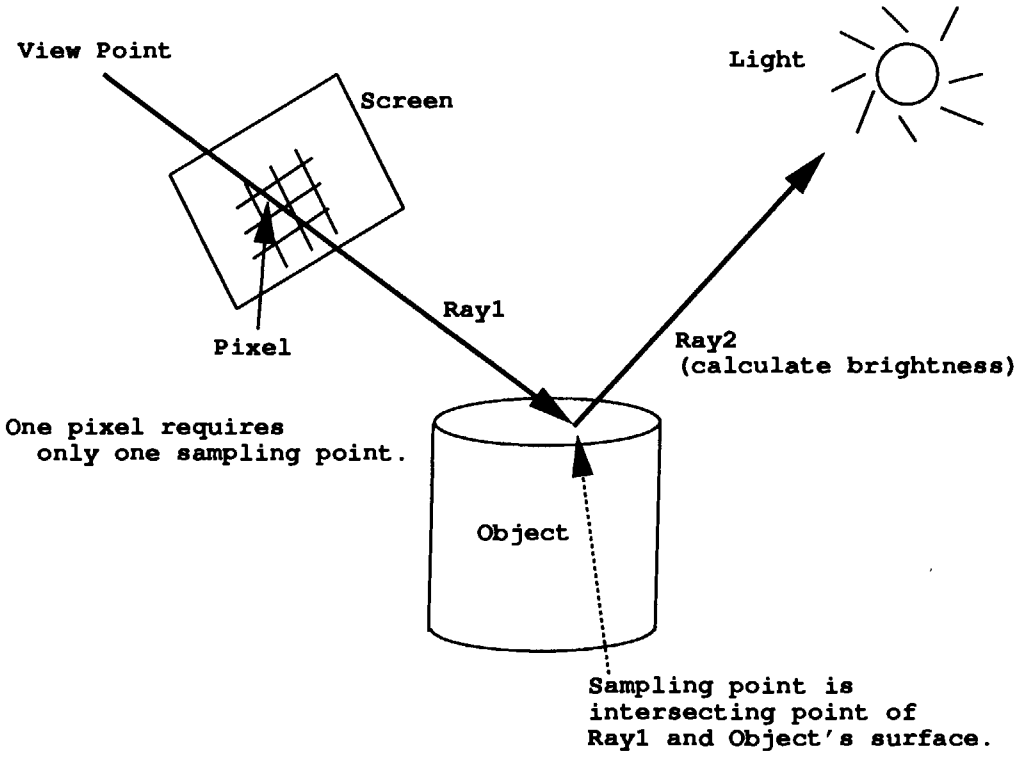


Fig. 3.2 Principle of ray tracing.

3.2 ボリュームレンダリングのアルゴリズム

Fig.3.3にボリュームレンダリング・モジュールのアルゴリズムを示す。ボリュームレンダリング・モジュールは実時間可視化システムからサブルーチン形式で呼び出されることになっ

ているため、可視化すべきデータの入力や前処理についてはこのモジュールでは考慮していない。また、実時間可視化システムは解析計算を行うコードに埋め込まれるため、ボリュームデータの形式は解析計算に用いられる計算格子上に計算結果の物理量が格納されたものである。

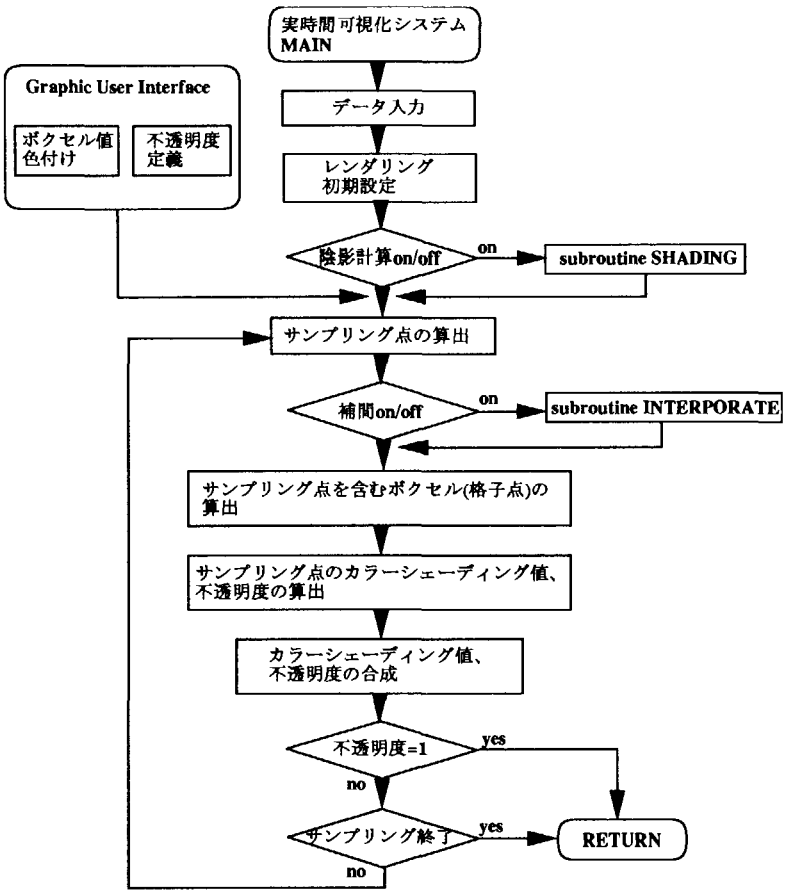


Fig. 3.3 Algorithm of the volume rendering module.

ボリュームデータへの不透明度の定義は、実時間可視化システムのボリュームレンダリング用 GUI(Graphic User Interface) によって対話形式でユーザが定義することになる。

シェーディングとは、立体感の表現に不可欠な濃淡づけのことである。例えば、球を可視化する際、斜め上方から光線を与えると光が当たっている部分は明るく、光の当たらない部分は暗くなるはずである。このように光源と物体表面の向き（法線方向）の関係から生じる物体上の濃淡（明暗）をつける（計算する）ことをシェーディングという。球をシェーディングすること無しに可視化すると、その結果可視化された物体が円なのか、球なのか判別不能になってしまう。したがって、３次元物体の可視化の際にシェーディングは非常に重要となる。

一般のＣＧ手法においては、シェーディングは物体表面の法線ベクトルと光源からの光線の方向ベクトルの内積を用いて計算される。ところが、ボリュームレンダリングでは物体表

面といった概念が存在しない。ゆえに可視化対象であるボリューム内に表面が存在するかどうかはわからないのである。そこで、各ボクセルについて、周囲のボクセルのデータ値（物理量）から求めたデータ値の勾配値（gradient）を当該ボクセルにおける法線ベクトルとして、Phongのモデル[6]を用いてシェーディングを行っている。

また、シェーディングと同時にボリューム内のデータ値に対応した色づけも行っている。ここで、温度が100℃なら赤色、50℃なら黄色、0℃なら青色などといったデータ値に対する色の対応づけは、不透明度の定義同様実時間可視化システムのGUIによって対話形式でユーザが行う。

したがって、色情報を含めたシェーディング（カラーシェーディング）は、まず法線ベクトルを3.1式で求め、法線ベクトル$N(x_i)$と光源方向ベクトルLの内積値からシェーディング値を求め、これに各ボクセルのデータ値に与えた色情報$C_k(x_i)[k=R,G,B]$との積によりカラーシェーディング値を求める。

$$N(x_i) = \frac{gradf(x_i)}{|gradf(x_i)|} \tag{3.1}$$

$$S(x_i) = h(N(x_i)\cdot L) * C_k(x_i) \tag{3.2}$$

ここで、hは拡散反射によるシェーディング関数である。鏡面反射まで考慮するとさらに複雑な式となるが、実用上拡散反射だけ考慮すれば十分である[4]。

レイキャスティングでは、スクリーン（の各画素）から任意角で視線（レイ）を発し、3次元空間において定義されているボクセルとの通過に伴う演算により、最終的にスクリーン面上へ投影する画素値を求める。

具体的には、各ボクセルに対し視線が入射すると、各ボクセルにあらかじめ与えられている不透明度$\alpha(x,y,z)$とカラーシェーディング値$S(x,y,z)$との乗算によりボクセルを通過した光（色情報）が求まる。しかし、視線は一般にボクセルに対し任意角で入射するため、常に（ボクセル値を持つ）格子点と交差するとは限らない。このため、より正確な不透明度とカラーシェーディング値を表示させるためにはFig.3.4に示すように格子点のリサンプリングが必要になる。

リサンプリングでは、近傍の格子点に与えられている$\alpha(x,y,z)$と$S(x,y,z)$を補間演算（8点近傍補間など）することによって新しい不透明度$\alpha(u,v,w)$とカラーシェーディング値$S(u,v,w)$を求める（u,v,wはリサンプリング座標）。

次にレイキャストにより画素値を求める。具体的には、視線が当たるボクセルについて、前から入ってきた色情報と、自らが発する（光源に照らされて反射する）色情報とを合成しながら、視線に沿って次々とボクセルに渡していき、最後のボクセルから出てきた色情報が生成しようとしている画像の画素値となる[7]。すなわち、

$$S_{out}(u,v,w) = S_{in} + S(u,v,w)\alpha(u,v,w)(1-\alpha_{in}) \tag{3.3}$$

$$\alpha_{out} = \alpha_{in} + \alpha(u,v,w)(1-\alpha_{in}) \tag{3.4}$$

ここで、S_{in}は前から来たカラーシェーディング値、α_{in}は前から来た不透明度、S_{out}は次のボクセルに渡されるカラーシェーディング値、α_{out}は次のボクセルに渡される不透明度である。

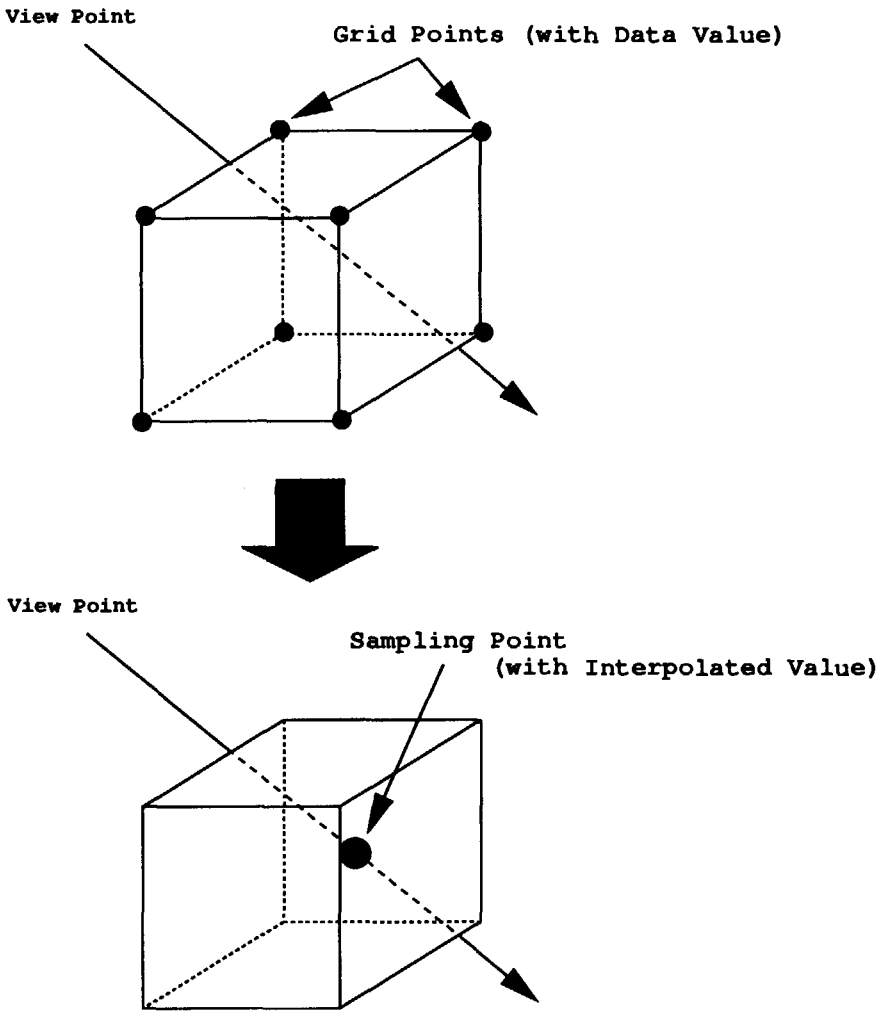


Fig. 3.4 Resampling for ray-casting.

3.3、3.4式により、視線がボクセルに当たるごとに視線のもつカラーシェーディング値及び不透明度にそのボクセルのカラーシェーディング値・不透明度を加えて更新し、途中で不透明度が1（不透明）になった場合は、その視線の計算を終了する。

4. ボリュームレンダリング・モジュールによる画像生成と計算処理時間

4.1 計算時間とレンダリングの高速化

これまでボリュームレンダリングの処理手順について記述してきたとおり、ボリュームレンダリングにおいてはスクリーンから発せられる視線ごとに交差する全てのボクセルとの色情報の合成計算を繰り返し行うため、一般のＣＧ手法に比べて計算時間が膨大となる。

ここで、色情報の計算もさることながら、各視線がどの順番でどのボクセルと交差しているのかを探索するための計算時間が無視できない。これはBFC格子などの場合は正方直交格子のようにサンプリング点を含む格子が計算式から算出ができず、探索しなくてはならないためである。この探索を極力効率的に行うことがレンダリング全体の計算時間の短縮のために重要であり、そのためには、解析計算に用いられている計算格子の形状の特性を利用することが重要となる。

今回開発したボリュームレンダリング・モジュールは、サンプリング点（視線上の任意の点）に最も近い格子点Pを見つけ出し、Pとサンプリング点の位置関係を考慮してPの前後のインデックスをもつ格子点を探ることでサンプリング点がどのボクセルに含まれるかを探索している。すなわち、構造格子ではインデックスjの隣の格子点は必ず$j-1$と$j+1$であるという特性を用いている。

このとき、サンプリング点の最近傍の格子点を毎回総探索していたのでは効率が悪い。そこでn番目のサンプリング点について探索する場合はn-1番目のサンプリング点の近傍のみを探索することで探索時間の削減を図っている。しかし、近傍のみを探索すると言っても、その必要かつ十分な探索範囲というのは、BFC格子の形状とレイキャスティングのサンプリング幅との兼ね合いによって常に変化する。

Fig.4.1に示すとおり、n-1番目のサンプリング点の近傍として、いくつの格子点（ボクセル）をチェックすればn番目のサンプリング点を含むボクセルが見つけられるかというのは格子点の粗密に依存しており、格子が粗い領域では比較的少数のボクセルの探索ですむが、格子が密になれば探索対象となるボクセルは増加することになる。

これに対処するために、２つの方法を用いてモジュールを作成し、その計算時間を比較した。１つは、格子形状に対し、サンプリング幅を十分小さくすることによってn-1番目のサンプリング点の最近傍であった格子点の隣だけを探索すれば（探索範囲を１に限定）最近傍点が見つかることを保証する方法。もう１つは、探索範囲を小さくとっておいてその範囲を探索し、見つかった近傍点が最近傍点でなかった場合は徐々に（発見的に）探索範囲を広げていくという方法である。両者のレンダリング処理時間を比較した一例をTable.4.1に示す。

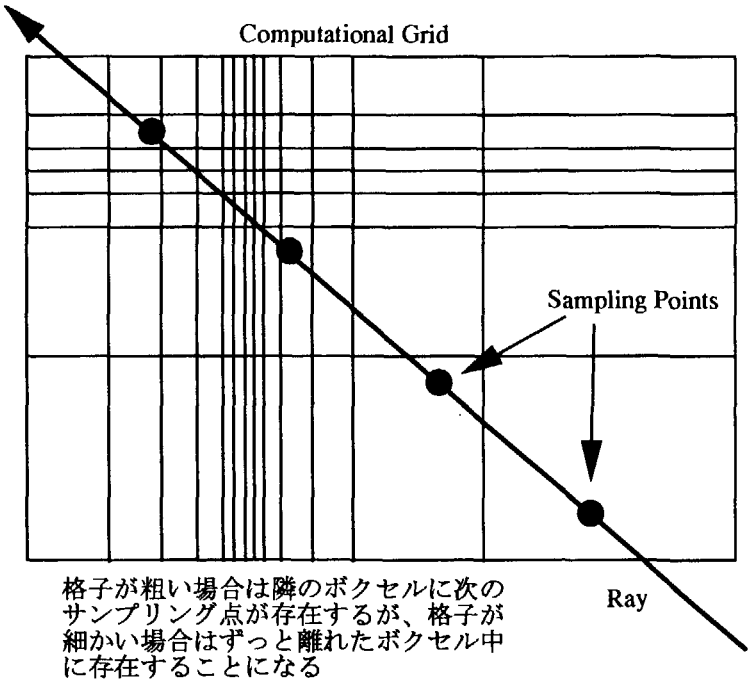


Fig. 4.1 Voxel search of ray casting.

Table.4.1は、格子数67x47x37で大気中の酸性雨原因物質の流動を計算した結果をNEC SX-4（1PE）を用いて可視化したケースの一例（平行投影、陰影なし、補間あり）である。

サンプリング幅を小さくして探索範囲を1に補償する方法は、総探索の場合に比べ16～20倍の高速化を達成しており、発見的に探索範囲を広げる方法では総探索に比べて24～33倍の高速化を達成している。

総探索の場合と発見的に探索範囲を広げる場合は1本の視線当たりのサンプリング数が81回であったのに対し、サンプリング幅を小さくして探索範囲を1に保証する場合は1視線当たり198回のサンプリングを要しており、このため計算時間の短縮率が低い。これは格子の大きさが不均一であることに起因し、最も小さい格子に合わせてサンプリング幅を決定しているためである。

なお、スクリーンの画素数が大きくなるに従い、可視化処理全体に対しレンダリング処理の占める割合が大きくなるため、スクリーンの画素数が大きくなるほどボクセル探索の高速化の効果は大きくなる。可視化システムとしての実用の際には、256x256あるいは512x512といった画素数が一般的であるので高速化の効果はもっと大きくなると期待される。

以上の結果を踏まえて、2つの方法のうちの後者、すなわち発見的に探索範囲を広げていく方法を採用することとした。

Table 4.1 CPU time for volume rendering.

可視化データ：大気中の物質の流動計算結果

計算格子数：67x47x37

使用計算機：NEC SX-4 (1PE)

備考：リサンプリングは 4 近傍線形補間

スクリーン画素数	ボクセル探索の手法		
	総探索の場合	サンプリング幅の調整による方法	発見的に範囲を広げる方法
32x32	145 秒	8.88 秒	6.06 秒
64x64	590 秒	27.7 秒	17.5 秒
128x128	30 分以上	102 秒	61.1 秒

4.2 ボリュームレンダリング画像

開発したボリュームレンダリング・モジュールを前述の実時間可視化システムに統合し、ボリュームレンダリング画像を作成した例を示す。

以下に示す画像は、前節で Table.4.1 によりレンダリング処理時間を示した例と同一であり、67x47x37 の計算格子数で、大気中（東アジア）の酸性雨原因物質の流動を NEC SX-4 を用いて計算し、その結果を可視化したものである。

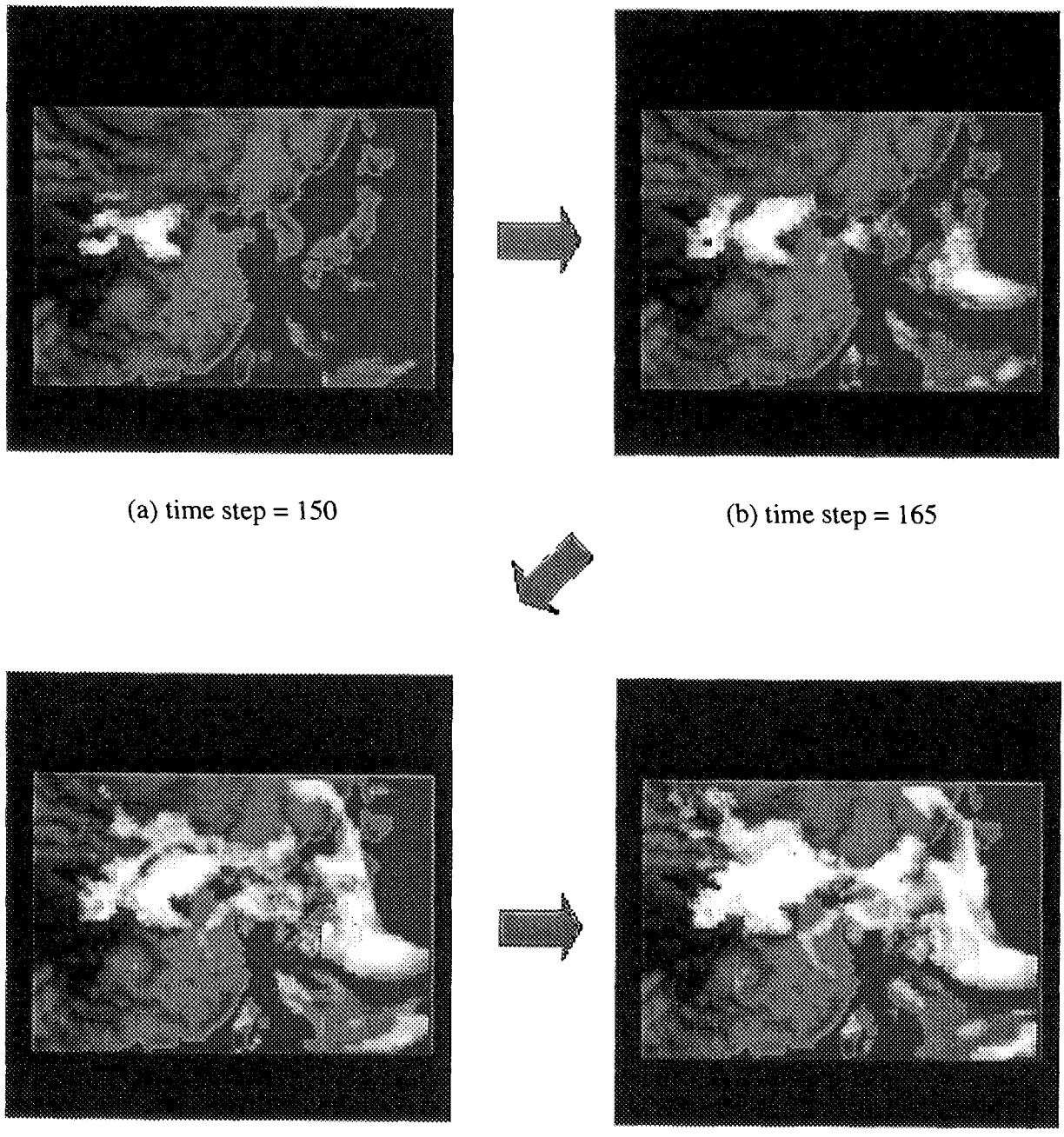
大気中の酸性雨原因物質の濃度分布をボリュームレンダリング画像で表示している。画像中の白い部分（不透明な部分）ほど物質の濃度が高いことを示している。

実時間可視化システムの他の可視化機能を使って描いた東アジアの地図の色情報を上記の計算結果のボリュームレンダリングの際に加味することによって、地図とボリュームレンダリング画像との結合を実現している。

Fig.4.2 のボリュームレンダリング画像は、256x256 画素の大きさで作成されており、ボリュームレンダリングに要する時間は CPU time で約 90 秒であった。この計算処理時間は、前節で記述した格子探索部分の高速化の効果は十分認められるものの、実時間可視化システムの可視化機能としてはより一層の高速化が必要なレベルと考えられる。

比較のために Fig.4.3 に同一のデータを等値面図により可視化した例を示す。Fig.4.3 では、白い領域は一定の濃度範囲の酸性雨原因物質の分布域を示しており、その拡散の様子はわかるが、物質の濃度分布まではわからない。

これに対し、ボリュームレンダリング画像は拡散の様子とともに、その濃度分布が一目瞭然である。



(a) time step = 150

(b) time step = 165

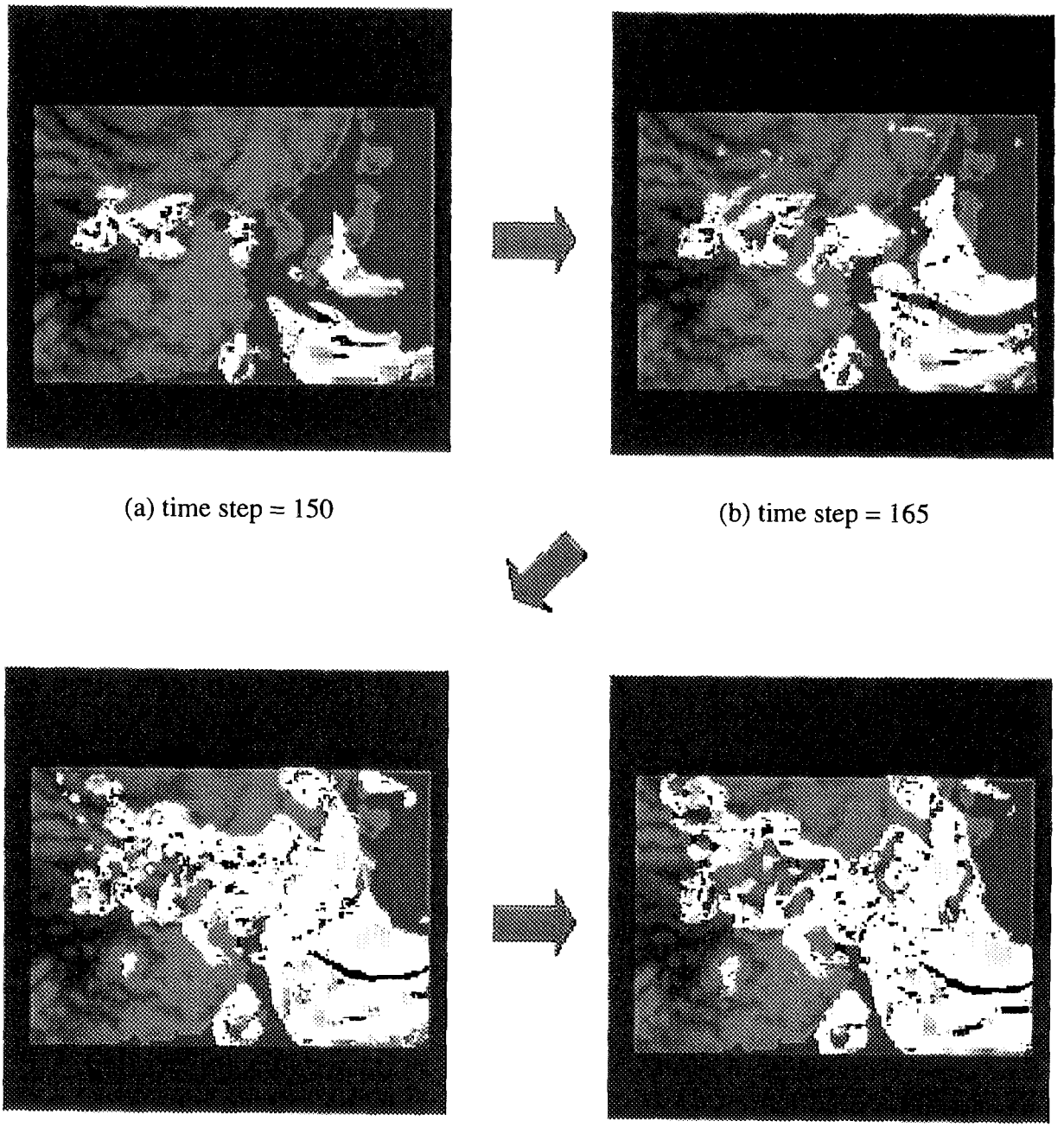
(c) time step = 180

(d) time step = 195

酸性雨の原因となる物質が西から東へ拡散していく様子と
その濃度分布がわかる。

Fig. 4.2 An example of volume rendering image.

This is a blank page.



(a) time step = 150

(b) time step = 165

(c) time step = 180

(d) time step = 195

酸性雨の原因となる物質が西から東へ拡散していく様子は
わかるが、内部の濃度分布まではわからない。

Fig. 4.3 An example of iso-surface image.

This is a blank page.

5. ボリュームレンダリングの並列化について

一般のＣＧ手法の高速化、例えばレイトレーシングなどの場合には、表示すべき物体に含まれる面（大抵の場合スクリーンに最も近い面のみ）との色情報の計算だけが有用（結果に反映される）であって、それ以外の面との計算は不必要なものとなる。ゆえに、バウンダリボリュームの採用などにより、結果に反映されない計算を事前に回避することが有効とされる。一方、ボリュームレンダリングは表面のみならずボリュームの内部構造まで半透明表示しようとするものであるから、原則的に全てのボクセルとの色情報計算が必要となる。前章では、サンプリング点を含むボクセルの探索において回避すべき計算を明らかにし、計算時間を短縮したが、色情報・不透明度の算出においては回避すべき不必要な計算というものは存在しない。

したがって、ボリュームレンダリングにおいては、どうしても多大な計算を速く行う方法が求められる。そこで、ボリュームレンダリングの高速化のための有効な方法の１つに並列処理が挙げられる。

並列処理とは、複数の計算プロセッサに同時並行に計算を実施させることにより、大規模な計算を高速に処理しようとするものである。そのためには並列計算機が必要となるが、もともと実時間可視化システムは並列計算機上での CFD(Computational Fluid Dynamics) 計算のための可視化ツールとして開発されたものであり、当然ボリュームレンダリング・モジュールについても並列処理を念頭に置いている。

レンダリング（レイトレースまたはレイキャスト）に関して並列処理を考慮する場合、視線の追跡をプロセッサ毎に割り当て、計算を並列実行する視線分割による並列処理が最も有力となる。これは、原理的にレンダリング計算が視線毎に独立であるという並列性の高さに起因する。原理的に依存関係のある計算を並列処理しようとしても、例えばＡという処理をした結果を踏まえてＢという処理を行わなければならない場合、Ｂの処理を担当するプロセッサが計算可能な状態にあってもＡの結果を待ってからでないとＢの計算をスタートできないということでは、並列処理を行うメリットがない。つまり、原理的に独立な計算を並列処理することが、並列処理による計算時間短縮の理想的な条件なのである。したがって、視線分割によるレンダリングは並列処理にとって理想的な計算の一つである。

しかし、実時間可視化システムにおけるレンダリング・モジュールの並列処理に関しては、レンダリングのアルゴリズムのみを考慮して並列処理の指針を決定するのでは不十分である。実時間可視化システムにおいては、解析計算とその結果の可視化が同一のプロセッサで連続して行われる（解析計算と可視化計算が独立していない）ため、解析計算のアルゴリズムをも考慮し、その計算結果が計算機の中のどこにどのように蓄積されるかを踏まえた上で可視化処理の並列化について論じる必要がある。

解析計算のアルゴリズムに目を移すと、とりわけ実時間可視化システムの当面の対象となる CFD の分野では、解析すべき領域を分割し、複数のプロセッサに割り当て同時並行に計算

させる方法、すなわち領域分割による並列処理がポピュラーである。そこで、本報告ではボリュームレンダリング・モジュールについて視線分割によって並列処理を行う場合と領域分割によって並列処理を行う場合のメリット・デメリットについて考察してみる。

5.1 視線分割による並列処理

視線分割による並列処理の概念図を Fig.5.1 に示す。

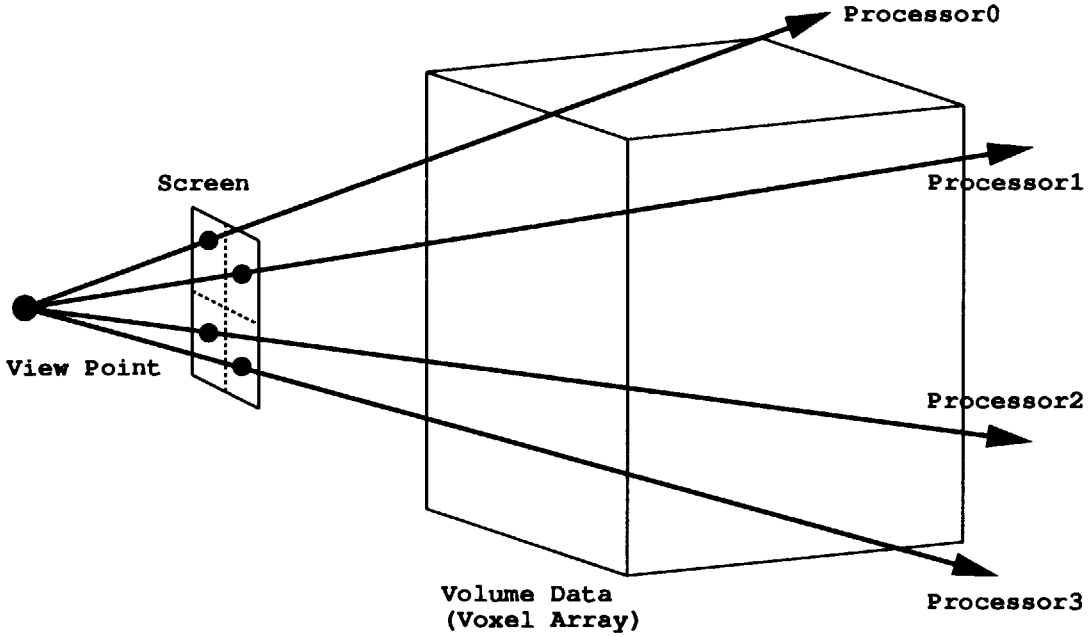


Fig. 5.1 Parallelization based on ray-decomposition.

視線分割の場合は、レンダリングの視線ごとの独立性を最大限に活用可能であり、スクリーンの画素ごとの並列処理が可能である。また、解析計算を各プロセッサでどのように分担したかとは無関係に視線追跡を行うことが出来る。したがって、レンダリングの計算付加に応じて各プロセッサに画素ごとの計算を分担させることが可能となり、レンダリングの付加を複数のプロセッサに均一に分散させることが可能となる。

一方、上記のメリットとは裏表の関係になるが、解析計算と画像生成を行うプロセッサが一致している必要がないため、解析計算結果を記憶しているプロセッサと画像生成を行うプロセッサとの間でデータ通信が必要となる。また、視線ごとに同時並行に計算を実施することになるので、複数のプロセッサが同時にあるプロセッサが記憶しているデータにアクセスしようとする（アクセス競合）可能性がある。この場合は、データの通信に待ち時間が発生するので、頻繁にアクセス競合が起こると高速計算の妨げになる。

Table.5.1 に視線分割による並列計算の特徴をまとめる。

5.2 領域分割による並列処理

領域分割による並列処理の概念図を Fig.5.2 に示す。

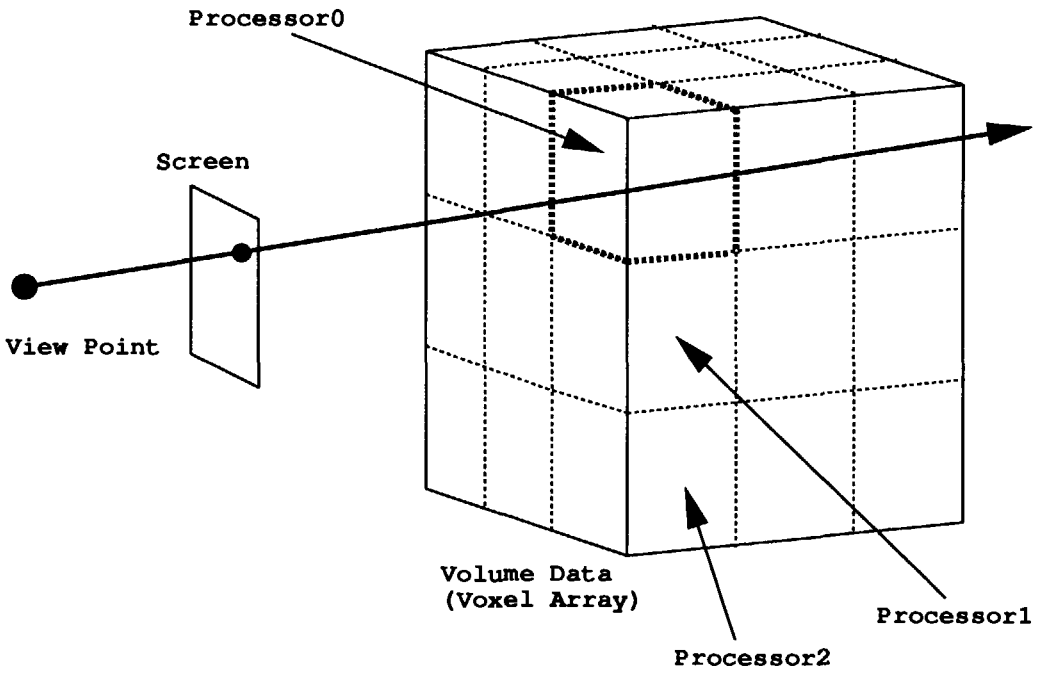


Fig. 5.2 Parallelization based on domain-decomposition.

領域分割の場合は、ある領域の解析計算を行ったプロセッサがその領域のレンダリングも担当することになる。このため、レンダリングに必要なボリュームデータは既に当該プロセッサ上に存在することになり、解析計算結果を他のプロセッサへ通信する必要がない。また、ボリュームデータに対するアクセス競合も発生しない。

逆に、解析計算とレンダリングが不可分となるため、解析すべき物理量の変動が激しい領域の計算を担当したプロセッサが引き続き物理量の変動が激しい領域の可視化を行うこととなり、プロセッサ間での付加が分散されず、特定のプロセッサに集中する可能性がある。また、分割されたボリューム領域ごとに生成された画像は、これを正しく画像合成して１枚の画像にまとめた後でなければ役に立たない。ボリュームレンダリング画像の合成は、単に当該画素の色データを足し合わせればよいというものではなく、視線との交差の順番（前後関係）を考慮し、演算する必要がある。画像の合成処理は単一のプロセッサ上で行わなければならず、この時にはプロセッサ間のデータ通信が不可避である。

ただし、解析計算と画像生成を行うプロセッサにしてみれば、結果を画像合成プロセッサに送信した後は次の計算が始められ、画像合成プロセッサは他のプロセッサが次の解析計算・画像生成をしている間に画像合成処理が実行できればよいので、解析計算・画像生成と画像合成の負荷バランスによっては、Fig.5.3 に示すような一種のパイプライン処理が形成でき、効率的に計算できる可能性がある。現状では、この点において領域分割による並列処理の方

が優位と考えられる。

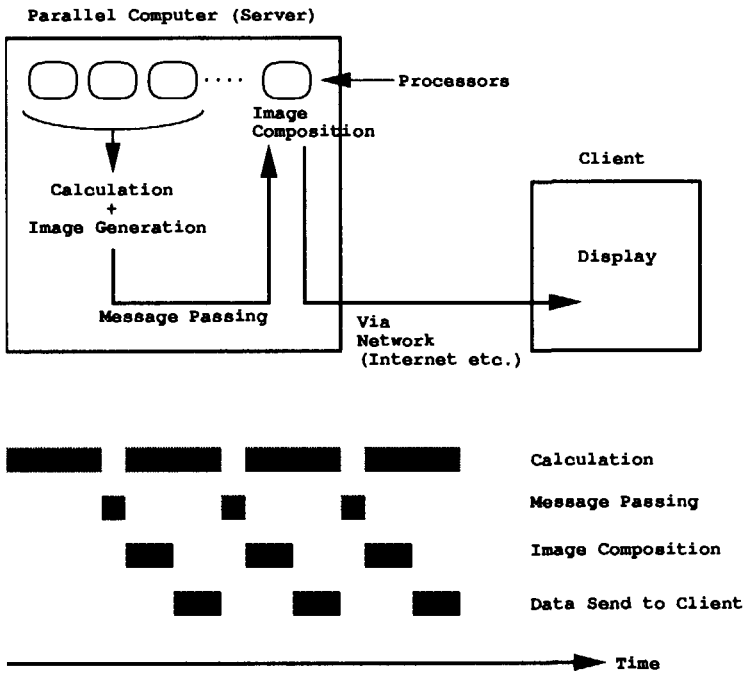


Fig. 5.3 Processing on parallel computer (server).

Table.5.1 に領域分割による並列計算の特徴をまとめる。

Table 5.1 Features of ray-decomposition parallel volume rendering and domain-decomposition parallel volume rendering.

	視線分割による 並列レンダリング	領域分割による 並列レンダリング
並列性	完全に並列処理可能	領域毎の画像生成は 並列に処理可能だが 領域毎の画像生成後 画像の合成が必要
プロセッサ間通信	各サンプリング点の 計算の都度に必要	領域毎の画像を合成 する際にのみ必要
データアクセス競合	可能性あり	なし
プロセッサ毎の負荷 の分散	可能	困難

6. おわりに

原研で開発してきた実時間可視化システム用にボリュームレンダリング・モジュールを新たに開発した。開発されたモジュールは、計算格子としてBFC格子が採用されている場合を強く意識して、その格子データを直接可視化することができる。

その際の計算時間、とりわけサンプリング点の近傍点探索の時間を短縮するアルゴリズムを試作し、レンダリング処理時間を1/30以下に短縮した。

しかしながら、それでも実時間可視化システムに組み込んで実用に耐えるには、一層の高速化が必要であることがわかった。このため、並列処理による計算時間の短縮についても考察し、今後はこれに基づいてさらなる高速化を行う。

また、現時点では、ボリュームレンダリング・モジュールは直交格子、BFC格子等の構造格子にのみ対応しており、有限要素法による構造解析等に利用される非構造格子への対応についても今後の課題となっている。

謝　　辞

ボリュームレンダリング・モジュールの開発及び実時間可視化システムへの実装にあたり、多大な御協力を賜ったNEC情報システムズの松本秀樹氏ならびにNEC C&Cメディア研究所の武井利文氏に深謝申し上げます。

参考文献

[1] 村松ほか：並列計算機上での流体解析のための実時間可視化システム, 計算工学講演会論文集, vol.2, no.1, pp.109-112(1997).

[2] 村松ほか：並列計算機上での流体解析のための実時間可視化システムの開発, JAERI-Data/Code 98-014, 日本原子力研究所 (1998).

[3] 大谷ほか：実時間可視化システム用ボリュームレンダリングモジュールの開発, 計算工学講演会論文集, vol.4, no.1, pp.329-332(1999).

[4] テレビジョン学会編：３次元ＣＧ, オーム社 (1994).

[5] 今間：ＣＧ入門セミナー, 日経ＢＰ社 (1998).

[6] Phong B.T. : Illumination for Computer Generated Pictures, *Comm. ACM*, vol.18, no.8, pp.311-317(1975).

[7] Porter T. and Duff T. : Computing Digital Images, *Computer Graphics(SIGGRAPH'84 Proceedings)*, vol.18, no.3, pp.253-259(1984).

[8] Hsu, W. M. : Segmented Ray Casting for Data Parallel Volume Rendering, *1993 Parallel Rendering Symposium Proceedings*, pp.7-14(1993).

[9] 對馬ほか：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, no.7, pp.1709-1718(1995).

[10] Ma, K.-L. et al. : A Data Distributed, Parallel Algorithm for Ray-Traced Volume Rendering, *1993 Parallel Rendering Symposium Proceedings*, pp.15-22(1993).

国際単位系 (SI) と換算表

表 1　SI 基本単位および補助単位

量	名称	記号
長さ	メートル	m
質量	キログラム	kg
時間	秒	s
電流	アンペア	A
熱力学温度	ケルビン	K
物質量	モル	mol
光度	カンデラ	cd
平面角	ラジアン	rad
立体角	ステラジアン	sr

表 2　SI と併用される単位

名称	記号
分，時，日	min，h，d
度，分，秒	°，′，″
リットル	l，L
トン	t
電子ボルト	eV
原子質量単位	u

1 eV = 1.60218 × 10^{-19} J

1 u = 1.66054 × 10^{-27} kg

表 5　SI 接頭語

倍数	接頭語	記号
10^{18}	エクサ	E
10^{15}	ペタ	P
10^{12}	テラ	T
10^{9}	ギガ	G
10^{6}	メガ	M
10^{3}	キロ	k
10^{2}	ヘクト	h
10^{1}	デカ	da
10^{-1}	デシ	d
10^{-2}	センチ	c
10^{-3}	ミリ	m
10^{-6}	マイクロ	μ
10^{-9}	ナノ	n
10^{-12}	ピコ	p
10^{-15}	フェムト	f
10^{-18}	アト	a

表 3　固有の名称をもつ SI 組立単位

量	名称	記号	他の SI 単位による表現
周波数	ヘルツ	Hz	s^{-1}
力	ニュートン	N	m·kg/s^2
圧力，応力	パスカル	Pa	N/m^2
エネルギー，仕事，熱量	ジュール	J	N·m
工率，放射束	ワット	W	J/s
電気量，電荷	クーロン	C	A·s
電位，電圧，起電力	ボルト	V	W/A
静電容量	ファラド	F	C/V
電気抵抗	オーム	Ω	V/A
コンダクタンス	ジーメンス	S	A/V
磁束	ウェーバ	Wb	V·s
磁束密度	テスラ	T	Wb/m^2
インダクタンス	ヘンリー	H	Wb/A
セルシウス温度	セルシウス度	℃	
光束	ルーメン	lm	cd·sr
照度	ルクス	lx	lm/m^2
放射能	ベクレル	Bq	s^{-1}
吸収線量	グレイ	Gy	J/kg
線量当量	シーベルト	Sv	J/kg

表 4　SI と共に暫定的に維持される単位

名称	記号
オングストローム	Å
バーン	b
バール	bar
ガル	Gal
キュリー	Ci
レントゲン	R
ラド	rad
レム	rem

1 Å = 0.1 nm = 10^{-10} m

1 b = 100 fm^2 = 10^{-28} m^2

1 bar = 0.1 MPa = 10^5 Pa

1 Gal = 1 cm/s^2 = 10^{-2} m/s^2

1 Ci = 3.7 × 10^{10} Bq

1 R = 2.58 × 10^{-4} C/kg

1 rad = 1 cGy = 10^{-2} Gy

1 rem = 1 cSv = 10^{-2} Sv

（注）

1. 表 1 - 5 は「国際単位系」第 5 版，国際度量衡局 1985 年刊行による。ただし，1 eV および 1 u の値は CODATA の 1986 年推奨値によった。
2. 表 4 には海里，ノット，アール，ヘクタールも含まれているが日常の単位なのでここでは省略した。
3. bar は，JIS では流体の圧力を表わす場合に限り表 2 のカテゴリーに分類されている。
4. EC 閣僚理事会指令では bar，barn および「血圧の単位」mmHg を表 2 のカテゴリーに入れている。

換算表

力	N(=10^5 dyn)	kgf	lbf
	1	0.101972	0.224809
	9.80665	1	2.20462
	4.44822	0.453592	1

粘度　1 Pa·s(N·s/m^2) = 10 P(ポアズ)(g/(cm·s))

動粘度　1 m^2/s = 10^4 St(ストークス)(cm^2/s)

圧力	MPa(=10 bar)	kgf/cm^2	atm	mmHg(Torr)	lbf/in^2(psi)
	1	10.1972	9.86923	7.50062 × 10^3	145.038
	0.0980665	1	0.967841	735.559	14.2233
	0.101325	1.03323	1	760	14.6959
	1.33322 × 10^{-4}	1.35951 × 10^{-3}	1.31579 × 10^{-3}	1	1.93368 × 10^{-2}
	6.89476 × 10^{-3}	7.03070 × 10^{-2}	6.80460 × 10^{-2}	51.7149	1

エネルギー・仕事・熱量	J(=10^7 erg)	kgf·m	kW·h	cal(計量法)	Btu	ft·lbf	eV
	1	0.101972	2.77778 × 10^{-7}	0.238889	9.47813 × 10^{-4}	0.737562	6.24150 × 10^{18}
	9.80665	1	2.72407 × 10^{-6}	2.34270	9.29487 × 10^{-3}	7.23301	6.12082 × 10^{19}
	3.6 × 10^6	3.67098 × 10^5	1	8.59999 × 10^5	3412.13	2.65522 × 10^6	2.24694 × 10^{25}
	4.18605	0.426858	1.16279 × 10^{-6}	1	3.96759 × 10^{-3}	3.08747	2.61272 × 10^{19}
	1055.06	107.586	2.93072 × 10^{-4}	252.042	1	778.172	6.58515 × 10^{21}
	1.35582	0.138255	3.76616 × 10^{-7}	0.323890	1.28506 × 10^{-3}	1	8.46233 × 10^{18}
	1.60218 × 10^{-19}	1.63377 × 10^{-20}	4.45050 × 10^{-26}	3.82743 × 10^{-20}	1.51857 × 10^{-22}	1.18171 × 10^{-19}	1

1 cal = 4.18605 J（計量法）

= 4.184 J　（熱化学）

= 4.1855 J　（15 °C）

= 4.1868 J（国際蒸気表）

仕事率　1 PS（仏馬力）

= 75 kgf·m/s

= 735.499 W

放射能	Bq	Ci
	1	2.70270 × 10^{-11}
	3.7 × 10^{10}	1

吸収線量	Gy	rad
	1	100
	0.01	1

照射線量	C/kg	R
	1	3876
	2.58 × 10^{-4}	1

線量当量	Sv	rem
	1	100
	0.01	1

（86 年 12 月 26 日現在）

実時間可視化システムのためのボリュームレンダリング・モジュールの開発