修士論文

ReVolver/C40 を用いた高精細リアルタイム可視化システムの実装

指導教官　　富田 眞治 教授

京都大学大学院情報学研究科
修士課程通信情報システム専攻専攻

山内 聡

平成14年2月8日

ReVolver/C40 を用いた高精細リアルタイム可視化システムの実装

山内 聡

内容梗概

我々はこれまでボリュームレンダリング専用並列計算機 *ReVolver*/C40 を開発・実装してきた。*ReVolver*/C40 ではボリュームレンダリングを、視線生成、ピクセル値の計算、シェーディングの 3 つの部分に分けそれぞれ専用基板を実装し、並列処理を行い高速描画を実現している。この *ReVolver*/C40 の手法の画質による評価と、更なるピクセル値計算の高速化のための様々な手法の実装と評価を行った。

ReVolver/C40 を実装した結果として、*ReVolver*/C40 のアーキテクチャはデータサイズに対してスケーラビリティがあることが分かった。そのため我々は、このスケーラビリティを生かし、さらに大きなサイズの可視化に対応する新たな可視化システムを提案した。このシステムでは *ReVolver*/C40 では実現が難しかった科学技術計算などのシミュレーションを実行するデータ可視化系との連携を前提としている。このとき、対象となるデータサイズは 4096^3 のボリュームデータを 2046^2 のスクリーンに $30fps$ で表示することを目標とた。

このような目標ではボリュームデータの転送にかかる時間がボトルネックとなる。そのため我々のシステムは、ネットワークを用いることなく高速に転送を行うべく、PC クラスタでシミュレーション等を実行し、PCI バスの付加するボードで *ReVolver*/C40 のアーキテクチャを応用した可視化専用ハードウェア実装したもので可視化を実行する構成をとる。この新システムの実現には、既存のハードウェアを用いて 128 台程度のシステムで実現可能であることがわかった。

この新システムの実装のテストとしてデータの並列転送に関する評価や必要となるソフトウェアの実装を行うために、*ReVolver*/C40 の 一部である PCS と PC クラスタを接続する PCI 通信ボードを実装し PC クラスタでシミュレーションを実行し PCS にデータを並列に転送し可視化を行うプロトタイプシステムを実装しこの並列転送の評価を行った。

Implementetion of a Real-Time Visualization System withReVolver/C40

Satoshi YAMAUCHI

Abstract

We have been developing Parallel Volume Rendering Machine '*ReVolver*/C40'. This machine is composed of with three independent stages. The Ray Casting Stage (RCS) creates view data from view point and screen coordinates. The Pixel Calculation Stage (PCS) calculates voxels addresses from view data generated by the Ray Casting Stage to get their values which are needed to calculate pixel values (colors and transparent values). The Shading Stage (SS) shades images generated by the Pixel Calculation Stage and displays them on the CRT. All these stages are controlled by the System Control Unit (SCU). The System Control Unit receives commands and data from the host machine, and transmits data and programs to each stage. The System Control Unit also provides a user interface. It was found that the Pixel Calculation Stage determines the speed of image generation. So we implement and evaluate several high speed method of Pixel Calculation, and we evaluate *ReVolver*/C40 in quality of picture.

Then we present a concept of real-time parallel volume rendering system for scientific visualization of 4000^3 volumetric datasets based on the scalable architecture of *ReVolver*/C40. In this system, a visualizer of volume rendering must cooperate with a producer that performs scientific simulation. Our goal is to design a new system that can generate images of 2048^2 pixels from 4000^3 voxels volume in 30 flames / second.

Because of this large-scale of volume data, the bottle neck of this new system is the translation time of this data. We construct this system with PC cluster and special volume rendering hardware in PCI slot. Using Internal bus of PC, we can transmit volume data in short time. To evaluate the effect of parallel transmissionin the real-time scientific visualization environment, we implement its prototype system with a part of *ReVolver*/C40.

ReVolver/C40 を用いた高精細リアルタイム可視化システムの実装

目次

第1章　はじめに

ボリュームレンダリングは、ボクセルと呼ぶ単位立方体で構成された3次元空間であるボリューム空間を、2次元のスクリーンに投影して画像を得る技法で、内部情報を持つデータの可視化が行えることから、物体の表面だけでなく、物体の内部や背後の状況も画像として表現することができる。

この技法は、従来医療画像分野において人体内部の可視化や、スーパーコンピュータなどの高速な計算機を利用した科学技術計算の結果の可視化による解析手法としても重要視されており、当研究室ではボリュームレンダリングを高速に実行できる専用並列計算機 *ReVolver*/C40 を開発、実装してきた。*ReVolver*/C40 では、512^3 のボリュームデータを実時間で可視化することを目標としていた。

近年では、地球シミュレータを代表とする大規模シミュレーションシステムが実用化されつつあり、それに伴ない膨大な量のデータが蓄積され、4000^3 程度の大規模な3次元データの解析を支援する3次元データ可視化システムが必要となりつつある。

ReVolver/C40 は実装したハードウェアデバイスの制限から、512^3 までボリュームデータの可視化にしか対応できないが、スケーラブルなアーキテクチャを利用しているため採用するデバイスを変更することで更に大規模なボリュームデータの可視化にも対応できる。我々はこの *ReVolver*/C40 のスケーラブルなアーキテクチャを利用して更に大規模なデータの可視化に対応する新たなシステムを実現を目指している。

科学技術計算の結果の可視化に関しては、*ReVolver*/C40 を含めた従来の可視化システムでは、最終結果の可視化に主眼がおかれているため、計算の途中経過を観測・解析して継続の判断基準とすることは難しく、あまり効率を得られない。そのため、シミュレーション等の科学技術計算をリアルタイムに可視化するシステムが必要となる。

そこで本稿では、シミュレーションによって得られる 4000^3 規模の3次元データをリアルタイムに可視化し解析・診断を支援するシステムを提案し、これまで開発を行ってきた *ReVolver*/C40 を用いたプロトタイプシステムの実装について述べる。

第2章　ReVolver/C40 概要

既に我々は、ボリュームレンダリング専用並列計算機 [1] の開発を行っており、その成果として *ReVolver*/C40[2] と呼ぶプロトタイプハードウェアを開発しその評価を行ってきた。今後はその *ReVolver*/C40 のデータサイズに対してスケーラブルなアーキテクチャを利用し更に大規模なデータの可視化に対応する新システムの実現を目指している。以下に *ReVolver*/C40 の概要としてボリュームレンダリングとは何か、特徴、どのようにボリュームレンダリングを処理しているか、構成を示す。

2.1　ボリュームレンダリング

ボリュームレンダリングは、ボリューム空間からサーフェイスを算出してレンダリングを行うサーフェイスレンダリングとは異なり、ボリュームの色や透明度を直接スクリーンに射影する手法である [3]。ボリュームレンダリングにおいて、可視化の対象となる空間をボリューム空間と呼ぶ。ボリューム空間には、科学技術計算の結果や自然現象などのデータセットが半透明のボリュームとして色や透明度などに変換されて置かれる。

ReVolver/C40 では、ボリューム空間は立方体であり、ある頂点を含む 3 辺がワールド座標系における各座標軸 (x,y,z 軸) の正の方向にあるように設定される。このボリューム空間を単位立方体に分割し、離散的にデータを与えたものをボクセルと呼び、各ボクセルには色や透明度の値 (ボクセル値) が与えられる。また、スクリーンはピクセルと呼ばれる単位正方形の集合として捉えられる。

2.2　ReVolver/C40 特徴

ReVolver/C40 は、医療画像生成だけでなく、科学技術計算の解析や流体の可視化をも行える専用並列計算機である。この目的を達成するためには、半透明ボリュームの表示、遠近法による画像生成、高速描画という 3 つの要件を満足する必要がある。ボリュームを半透明なものとして扱うには、図 1 のようにピクセル値を計算する際に視線と交差する無色透明でない全ボクセルを順にアクセスする必要がある。

また遠近法を採用する場合、任意方向の視線が発生するため、メモリアクセスが規則的ではなく、メモリバンクへのアクセス競合の可能性が高くなり、並

列処理への障害となる。ここでは、それらの問題を解決し、高速描画を行うために *ReVolver*/C40 が持っている特徴を述べる。

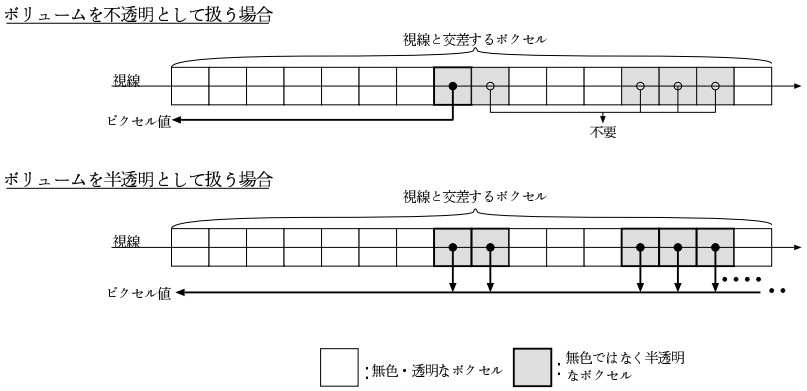

図1: 参照するボクセルの違い

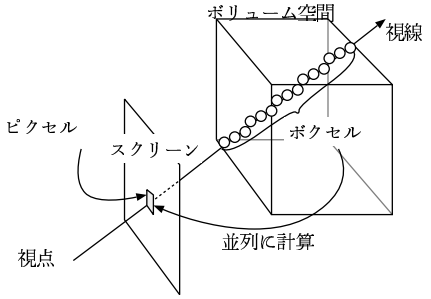

図2: レイキャスティング法

レイキャスティング法の採用　視点からスクリーン上のあるピクセルの中心へと向かうベクトルのことを視線ベクトルと呼び、視点を一端としピクセルの中心を通る半直線を視線と呼ぶ。図2に示すようにレイキャスティング法ではスクリーン上のピクセルごとに、視線がどのボクセルを通過してきたのかを追跡して、ピクセルの色や明るさを決定し、スクリーンに投影する。レイトレーシング法と同じ考え方であるが、反射や屈折のことは考えない。

サンプリング方法の単純化　従来のサンプリング方法では、視線の方向に対して等間隔にボクセルをサンプリングしてきた。この方法を視線等間隔サンプリング法と呼ぶ。

ReVolver/C40 では、視線ベクトルの成分の絶対値のうち、最大値をもつ座標軸を主軸と定め、この主軸の方向に対して等間隔にボクセルをサンプリングする方法を採用している。この方法を主軸等間隔サンプリング法と呼ぶ (図 3(b) 参照)。この方法により、次に述べるバンクコンフリクトが生じない。

ただこの方法では、視線が軸と平行でない時にサンプリング間隔が従来の視線等間隔サンプリング法 (図 3(a) 参照) と比べて大きくなるという性質がある。このサンプリング間隔の補正はピクセル値の計算時に行うことで対処する。

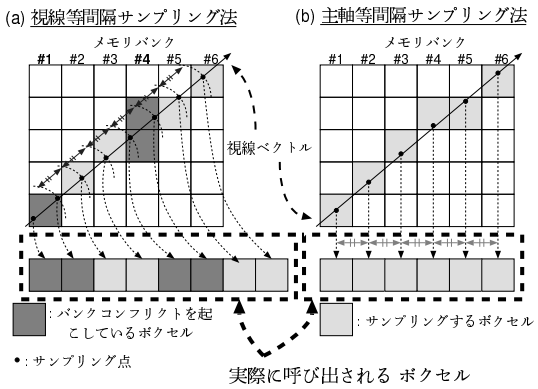

図 3: サンプリング方法の違い

バンクコンフリクトのないメモリ構成　主軸等間隔サンプリング法では、ある座標軸に垂直な slice にボリュームデータを分割してメモリバンクに格納すれば、その座標軸を主軸として持つ視線に関してはバンクコンフリクトフリーとなる。しかし、任意方向の視線の主軸が、この座標軸を主軸として持つとは限らない。そこで、x,y,z 軸のいずれにも垂直な slice が存在するように分割すれば任意方向の視線に対してバンクコンフリクトフリーとすることができる。

ReVolver/C40 の 3 次元メモリ構造では、図 4 のように各平面群に対して、平面に 1,2,$\cdots$,n と番号を付け、同じ番号を持つ平面のデータをその番号のメモリバンクの対応する領域に格納する。すなわち、ボリューム内の座標 (p,q,r) のボクセルは、メモリバンク p の X 領域、メモリバンク q の Y 領域、メモリバンク r の Z 領域の 3 箇所に格納されるので、ボリュームデータを

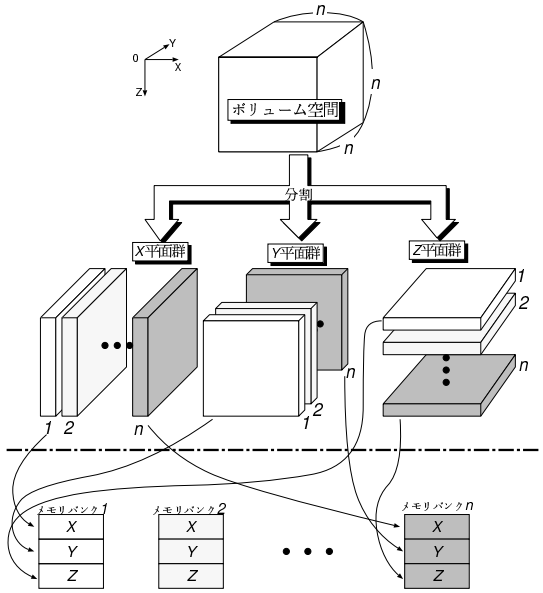

図4: ボリュームデータの格納方法

三重化し、格納されることになる。

この構成により、各平面群の平面は全て別のメモリバンクに格納されることになり、主軸等間隔サンプリングを行うことにより、視線が任意方向であっても、バンクコンフリクトが生じないことを保証する。

2.3 ReVolver/C40 処理

ボリュームレンダリング処理は、大きくわけて視線生成、ピクセル値計算、シェーディングの3つの処理にわけられる。*ReVolver*/C40ではこれらの3つの処理に特化した基板(RCS:レイキャスティングステージPCS:ピクセル値計算ステージ、SS:シェーディングステージ)を設計及び実装し、それらを接続することでボリュームレンダリング処理を高速に行うことを考えた。これらの*ReVolver*/C40が行う各処理について述べる。

2.3.1 視線生成処理

視線生成処理とはピクセル値計算に必要な視線に関するデータを生成する処理である。視線生成処理には以下に述べるような処理がある。

- 視線ベクトルの生成

  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それぞれワールド座標系で表したものを$\vec{S}_h$、$\vec{S}_v$、視点からスクリーンの左上(スクリー

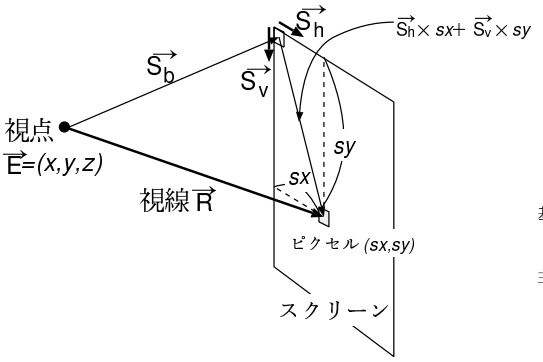

図 5: ビューデータの相互関係

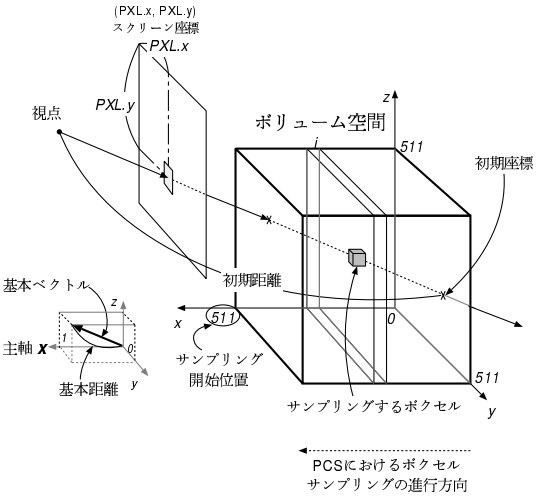

図 6: 視線データの相互関係

ン座標の原点) に向かうベクトルを $\vec{S}_b$ とする (図 5 参照)。ここで、スクリーン座標が $(\mathrm{s_x},\mathrm{s_y})$ であるピクセルに対する視線ベクトルを $\vec{R}$ とすると、$\vec{R}$ は次式 (1) で表される。

$$\vec{R} = \vec{S}_b + \mathrm{s_x} \times \vec{S}_h + \mathrm{s_y} \times \vec{S}_v \tag{1}$$

以下では視線データの相互関係を示した図 6 を適宜参照してもらいたい。

- 主軸及び主軸方向の判定

  ReVolver/C40 では主軸等間隔サンプリング法を採用している。この手法では、生成された視線ベクトルの成分の中で絶対値が最大であるものを選び、その座標軸を主軸と定義し、この主軸の方向に対して等間隔にサンプリングを行う。

  ReVolver/C40 で保持しているボリュームデータは、主軸が x, y, z 軸のいずれであっても対応できるように 3 重化され、それぞれの軸に対してバンク分けされている。そのため、ボクセルの読み出しの際には、主軸名が必要となる。また、視線ベクトルの主軸成分の符号によってピクセル値の計算方法が異なるので、主軸方向も必要である。$\vec{R}$ の成分の中で絶対値が最大のものが主軸となり、その符号が主軸方向となる。

- 基本ベクトルの計算

  視線ベクトルの主軸成分を 1 にしたものを基本ベクトル $\vec{B}_v$ と呼び、$\vec{R}$ の

主軸成分が$\mathrm{R_i}$ (i∈{ x,y, z}) であるとき、$\vec{B}_v$ は次式 (2) で表される。

$$\vec{B}_v = \frac{1}{\mathrm{R_i}}\vec{R} \tag{2}$$

また、シェーディング処理に用いられる基本ベクトルの大きさ (BLEN とする) は、$\vec{B}_v$ の非主軸成分を BV1、BV2 とすると、次式 (3) により求まる。

$$\mathrm{BLEN} = \sqrt{1 + \mathrm{BV1}^2 + \mathrm{BV2}^2} \tag{3}$$

- 視線とボリューム空間の交差判定

  ボリューム空間を全く通過しない視線については、ピクセル値計算を行う必要がないため、視線データにその旨を加えて伝える必要がある。そこで、視線がボリューム空間を通過するか否かの判定が必要となる。この判定は、以下の 2 点について行う。

  1. 視線方向による判定

     視点座標の主軸成分が負で主軸方向が負の場合、および主軸成分がボリューム空間の 1 辺の長さ (以下 VSIZE とする) より大きく主軸方向が正の場合は、視 線がボリューム空間と交差しない。

  2. ボリューム空間の主軸成分における端点の座標による判定

     まず、視線上の点で主軸成分の座標がボリューム空間の両端に等しい点 $\vec{R}_0$, $\vec{R}_{VSIZE}$ の座標をそれぞれ求める。ここでは、例として主軸が x である場合を考えて、$\vec{R}_0 = (0, \mathrm{R_{y0}}, \mathrm{R_{z0}})$, $\vec{\mathrm{R}}_{\mathrm{VSIZE}} = (\mathrm{VSIZE}, \mathrm{R_{yVSIZE}}, \mathrm{R_{zVSIZE}})$ とする。*ReVolver*/C40 では、視線の絶対値の最大成分を持つものを主軸としているため、視線の方向と主軸がなす角度は 45° 以下である。ボリューム空間は立方体であることを前提にしているので、視線がボリューム空間を通過するためには、$\vec{R}_0$ または $\vec{R}_{VSIZE}$ がボリューム空間の領域内になければならない。すなわち、

     $$0 \leq \{\mathrm{R_{y0}}, \mathrm{R_{z0}}\} \leq \mathrm{VSIZE} \tag{4}$$

     または

     $$0 \leq \{\mathrm{R_{yVSIZE}}, \mathrm{R_{zVSIZE}}\} \leq \mathrm{VSIZE} \tag{5}$$

     が成立しなければ、視線がボリューム空間を通過しないと判断できる。

- 初期座標、初期距離の計算

  ReVolver/C40で必要なボクセル値のアドレス計算の初期値となる初期座標(視線上の点のうち、主軸成分が0となる点で、非主軸成分をIN1,IN2とする)を$\vec{R}_0$、視点の位置ベクトルを$\vec{E}$、その主軸成分を$\mathrm{E_i}$とすると、入射距離(初期距離:ILENと呼ぶことにする)は次式(6)で表される。

$$\mathrm{ILEN} = |\vec{E} - \vec{R}_0| = |\mathrm{E_i}| \times \mathrm{BLEN} \tag{6}$$

- サンプリング開始位置の計算

  視点がボリューム空間内にある場合、視点より手前(スクリーンと反対側)のボクセルがピクセルに投影されないようにするため、そのボクセルを透明とみなすマスク処理を行う必要がある。そこで、図7のように、読み出すボクセルがどこから輝度計算に必要となるかをサンプリング開始位置MASKにより知ることができる。

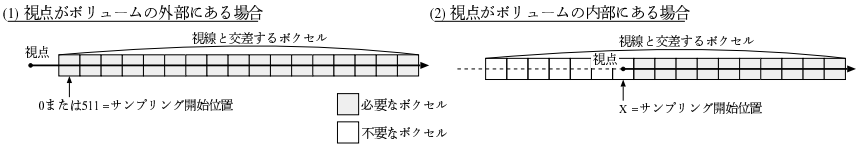

図7: サンプリング開始位置

ReVolver/C40では視線生成処理を視線生成ステージ(RCS)で行っている。

2.3.2 ピクセル値の計算

ReVolver/C40で行うピクセル値の計算は、1つのピクセルの輝度の計算を複数のプロセッサで処理するボクセル並列処理という方式を採用している。この方式はレイキャスティングアルゴリズムを視線方向に並列化したものである(図2参照)。

この方法では、スクリーン上の各ピクセルごとに発生する視線に沿って、視線と交差するボクセルを視点に近い順にサンプリングしていく。これを視線上のボクセルがなくなるまでくりかえし、ピクセル値を求める。この時のサンプリングされたボクセル数をNとすると、対応するピクセルの値は、以下の式(7)を評価した値となる。

$$pixel = \sum_{i=0}^{N}(1-\alpha_i)c_i \prod_{j=0}^{i-1}\alpha_j \tag{7}$$

ただし、ボクセル値v_tは、色c_tと透明度α_tを表すデータであり、サンプリングしたボクセルの値は視点に近い順から、$v_0, v_1, \cdots, v_{N-1}$とする。ここで、式(7)は式(8)のような漸化式に変形できる。

$$\begin{cases} C_k &= C_{k-1} + (1 - \alpha(v_k)) \cdot c(v_k) \cdot A_{k-1} \\ A_k &= \alpha(v_k) \cdot A_{k-1} \end{cases} \tag{8}$$

ただし、$c(v_i), \alpha(v_i)$はそれぞれボクセル値v_iを色、透明度に変換した値であることを示している。

サンプリングするボクセル数がNの場合、式(7)によって求められるピクセル値は式(8)のC_Nと同一の値となる。つまりC_Nを求めることによって、ピクセル値を求めることが出来る。

式(8)のC_k, A_{k-1}は$C_{k-1}, A_{k-1}, c(v_k), \alpha(v_k)$から求めることができ、$c(v_k), \alpha(v_k)$は$v_k$を変換した値であるから、結局$C_k, A_k$は$C_{k-1}, A_{k-1}, v_k$から求めることができる。

よって、2.2節で述べた主軸等間隔サンプリング法を用い、i番$(i = 0, 1, \cdots m)$のプロセッサにC_k, A_kの計算を行わせると、各プロセッサが2.2節で述べたメモリバンクの1つをローカルメモリとしてもち、プロセッサを一次元に接続した構成をとることができる。このときプロセッサ数は、ボリューム・データがN^3 voxelの場合N台となる。この構成により、プロセッサをパイプライン動作させることで、ピクセル値計算の高速化をはかることが出来る。

ところが視線の主軸成分が正(以下、「視線方向が正」と呼ぶ)の時と、視線の主軸成分が負(以下、「視線方向が負」と呼ぶ)の時とでは、処理の行う方向が逆となる。よって例えば視線方向が正のピクセル値の計算を行っている最中に、視線方向が負のピクセル値計算を行おうとすると、処理の衝突が起こり性能が低下する。

そこで、常に処理の方向を正とすることを、次のようにして実現した。

式(7)を式(9)に変形する。

$$C'_{k-1} = C'_k \cdot \alpha(v_{k-1}) + (1 - \alpha(v_{k-1})) \cdot c(v_{k-1}) \tag{9}$$

式(7)によって求められるピクセル値は、式(9)のC'_0と同一の値となる。式(9)のC'_{k-1}の値は$C'_k, c_{k-1}, \alpha_{k-1}$から求めることができるので、結局$C'_k, v_{k-1}$か

ら求めることができる。

よって、視線方向が負の場合に i 番のプロセッサに式 (9) の C'_{N-i} の計算を行わせると、ボリューム・データが $N^3 voxel$ とすると、$0, 1, \cdots, N$ 番のプロセッサの順で処理が行える。

以上より、視線方向が正の時は式 (8) を計算し、視線方向が負の時は式 (9) を計算することで、常に処理の方向を単方向とすることが可能である。

ReVolver/C40 ではこれらの処理をピクセル値計算ステージ (PCS) で行っている。

2.3.3 シェーディング処理

ReVolver/C40 ではこれらの処理をシェーディングステージ (SS) で行う予定であるが、現在、シェーディングに関しては何も行わず、計算されたピクセル値をそのまま表示している。

2.4 ReVolver/C40 の並列処理

ボリュームレンダリング処理は多大な計算量が必要であるため、並列処理が不可欠である。このため *ReVolver*/C40 では様々な規模、形の並列処理を行っている。

まず *ReVolver*/C40 ではフレーム単位で、画像生成、シェーディング、画像出力を独立に処理することによってマクロパイプラインを形成し、高速な表示を可能としている (図 8 参照)。

このパイプラインでは、画像生成、シェーディング、画像出力を独立に処理している。これらの 3 つの処理をフレームごとにパイプライン処理することで高速化を図っている (図 8 参照)。このなかで、画像生成は RCS と PCS が担当し、シェーディングと画像出力は SS が担当しており、それぞれが独立に処理を行っている。

さらに、画像生成においてはピクセル単位のパイプライン処理が行われている (図 8 参照)。また、2.3 節で述べた 3 つの処理はそれぞれの処理の特徴を考慮にいれ、以下に示すような様々な並列処理によって行われている。

視線生成処理での並列処理 ピクセル値計算処理の速度に見合った速度で視線データを生成するため、フレームをスキャンライン単位に分割して、8 台のプロセッサで並列処理を行っている。ピクセル値計算処理のためには視線データを逐次的に送らければいけないが、転送処理は他のプロセッサで

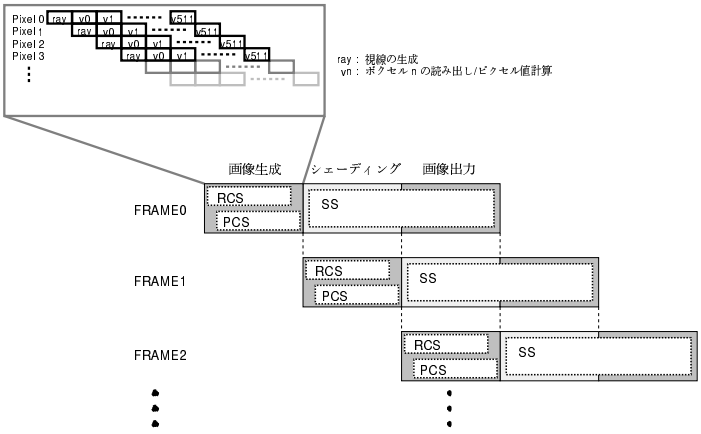

図 8: *ReVolver*/C40 のパイプライン構成

の視線生成処理に隠蔽される。

ピクセル値計算処理での並列処理　一つの視線に対するボクセル並列のピクセル値計算処理は一次元のパイプライン構成をとることによってその処理を並列化することが可能である。それぞれのプロセッサは視点に近いボクセルから順にピクセル値を計算し、自分の処理が終わると次のプロセッサに計算結果を渡す。プロトタイプではプロセッサは 128 台、最終構成のフルシステムでは 512 台使用する。

シェーディング処理での並列処理　シェーディング処理では、スクリーンを 16 個の二次元もしくは一次元のブロックに分割し、それぞれを一つのプロセッサに割り当て、シェーディング処理を並列に行う。

シェーディング処理を行うためには、ピクセル値計算処理の結果が必要である。しかし先ほど述べたように、ピクセル値計算処理は一次元のパイプライン構成をとっている。そこで、1) パイプラインから逐次的に送られてくるデータの受け取りとそれを 16 台のプロセッサで処理するための分配、2) シェーディング、3) シェーディングされたデータの収集と画像出力、という 3 段のパイプライン処理を行っている。

上述の通り視線生成処理とシェーディング処理は、処理すべきデータに対して空間的な並列処理を、またピクセル値計算処理は時間的な並列処理をそれぞれ行っている。

このように、視線生成処理とピクセル値計算処理との間、ピクセル値計算処理とシェーディング処理との間では並列処理の形態が異なるため、*ReVolver*/C40

では並列処理効率を低下させずにこれらの差異を吸収するための措置が講じられている。

2.5 ReVolver/C40 構成

ReVolver/C40 は図 9 に示すように、視線生成ステージ (RCS)、ピクセル値計算ステージ (PCS)、シェーディングステージ (SS) と呼ばれる 3 つのステージによって構成されており、これらが独立に処理を行うことによって、並列処理を実現している。この 3 つのステージをまとめて制御を行っているのが、システムコントロールユニット (SCU) である。

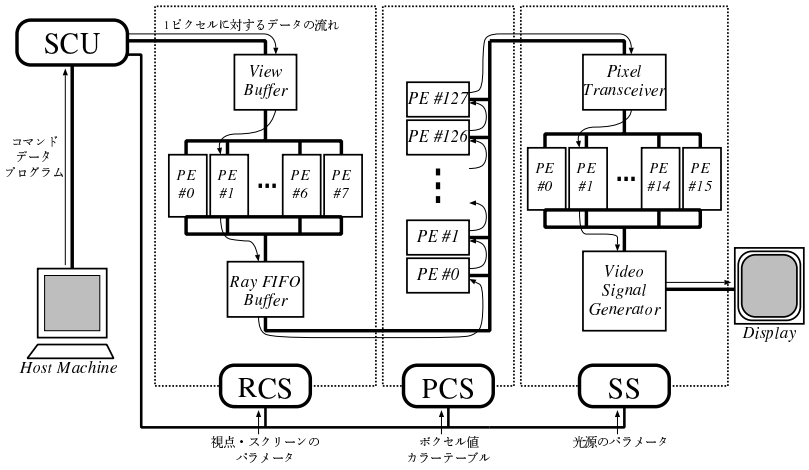

図 9: *ReVolver*/C40 の全体構成

ReVolver/C40 ではそれぞれのステージと SCU に対する基板を設計し、実装を行った。以下、本稿で必要となる、SCU,RCS,PCS の構成について述べる。

2.5.1 SCU の構成

SCU は、*ReVolver*/C40 システム全体の制御を行い、ホストマシンからの指示にしたがって、他の構成要素に対してコマンド/データを転送する役目を果たす。

SCU の構成要素は主に以下の 2 つである。

SCU メインボード　SCU メインボードは *ReVolver*/C40 の構成要素と接続するコネクタと、Xilinx 社の Field Programmable Gate Array:FPGA(XC4013E) で構成されており、ホストマシンの PCI スロットに接続する。

SCU サブボード　SCU サブボードは、*ReVolver*/C40 全体にクロック信号とリ

セット信号を供給する基板であり、主にクロックジェネレータ、Phase Locked Loop(PLL) などで構成されている。このボード上で PLL の設定を変えることによって、各基板に送るクロックの位相差をなくす。

2.5.2 RCS の構成

RCS は SCU から視点とスクリーンに関するパラメータ (ビューデータ) を受け取り、これらのデータと各ピクセルのスクリーン座標から視線ベクトルを求め、PCS におけるボクセルサンプリング及び SS におけるシェーディングに必要なデータ (視線データ) を出力する。RCS におけるこれらの処理を総称して視線生成処理と呼ぶ。

RCS の構成要素は主に以下の 3 つである。

Processing Element (PE) *ReVolver*/C40 では PE として、TI 社の TMS320C40 という Digital Signal Processor (DSP) を使用している。TMS320C40 は GLOBAL BUS および LOCAL BUS と名付けられた 2 つのバスをもつ。RCS では、GLOBAL BUS 側に View Buffer、LOCAL BUS 側に Ray FIFO Buffer が接続される。これにより、ビューデータの読み出しバスと視線データの転送バスが分離され、PE 間でのバス競合を軽減する。

View Buffer 全 PE と SCU で共有されるメモリで、SCU からのビューデータや PE のプログラムなどを保持する。View Buffer は 64Kwords の容量をもつ SRAM で構成され、シングルポート構成である。

Ray FIFO Buffer PE の生成した視線データを受け取る First-In First-Out(FIFO) メモリで、後の PCS でデータが二系統にわかれるため、AG-FIFO Buffer と、DSP-FIFO Buffer の二種類が存在する。視線データを、視線そのものに関する情報と、ピクセル値に関する情報に分割し、それぞれ AG-FIFO ならびに DSP-FIFO へ格納する。これにより、後述する PCS でのピクセル値計算とアドレス計算の並列処理が可能となる。

2.5.3 PCS の構成

PCS はボリュームデータを保持しているステージであり、RCS からの視線データを受け取り、これらのデータに基づいてその視線上にあるボクセル値を読み出す。ボクセル値は色と透明度に関するデータを表現しているが、*ReVolver*/C40 におけるボクセル値は 1byte のインデックス値である。よってこれを用いて属性テーブル [2] を参照し、ボクセルの色や透明度を求めてピクセル値計算を行う。そしてそのピクセル値を SS へ送る。

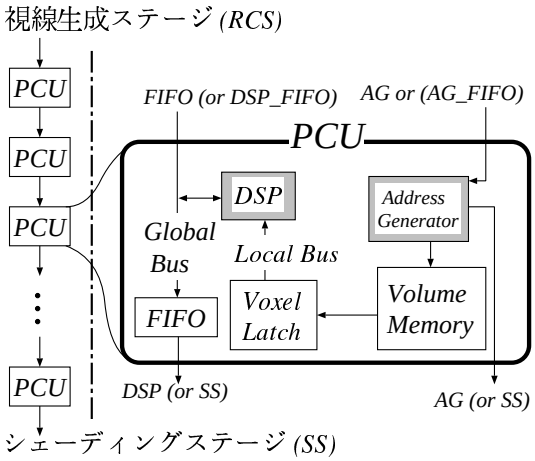

図 10: PCU の構成図

PCS はピクセル値計算ユニット (PCU) を 1 次元に接続させた構成となっている。1 枚の PCS 基板には、8 台の PCU を搭載し、この PCS 基板を直列接続することで、PCS を構成する。プロトタイプシステムの PCS は 16 枚の PCS 基板 (128 台の PE) で構成する。

PCU の構成要素は主に以下の 5 つである (図 10 参照)。

Processing Element (PE)　ピクセル値計算や、交差判定を行う。ボクセルデータアクセスを LOCAL BUS で、ピクセル値情報の送受信を GLOBAL BUS で行う。

Address Generator (AG)　サンプリングするボクセルのアドレス計算、ボクセル値の読み出し、隣接する AG 間の通信を行う。また通常の DRAM Controller としての処理や、PE の LOCAL BUS や Voxel Latch の制御も行う。AG は、Xilinx 社の FPGA(XC4010D) を用いて実現する。

FIFO Buffer　次段の PE へピクセル値情報の送信を行うためのバッファである。

3D Volume Memory　データを格納している DRAM である。*ReVolver*/C40 は最大で 512^3voxel のボリューム空間を扱い、1voxel が 1byte なので、データの 3 重化 [2] やプログラムの格納なども考慮して、4Mbyte の DRAM を用いる。

Voxel Latch　AG によってサンプリングされたボクセル値を PE によって読み出されるまで保持しておくための D-FLIP FLOP である。

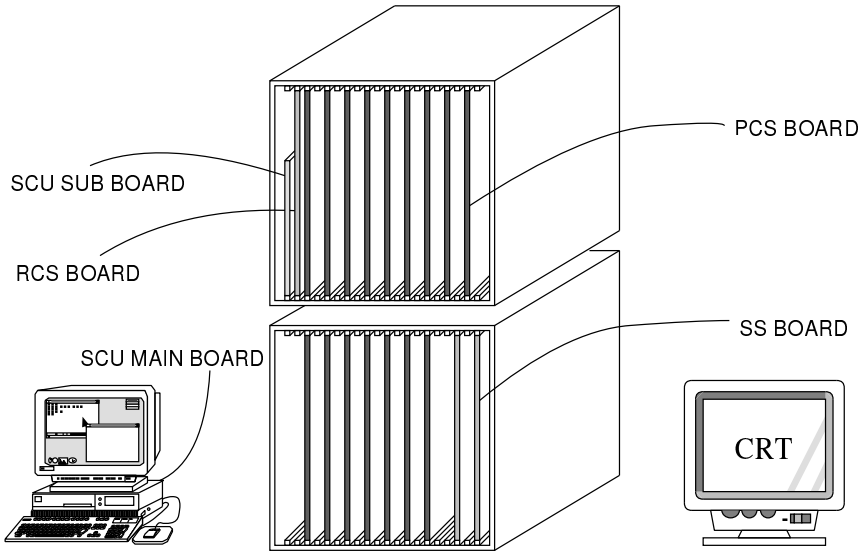

図 11: *ReVolver*/C40 の筐体

2.5.4 全体の構成

ReVolver/C40 を構成するためには、各ステージの基板はそれぞれ表1 に示した枚数が必要である。また、現状で動作させている *ReVolver*/C40-demo と呼ばれる形態に必要となる枚数も示す。

表 1: *ReVolver*/C40 の構成のために必要な各基板の枚数

基板名	基板枚数		
	最終構成	プロトタイプ	demo
SCU	1 組	1 組	1 組
RCS	1 枚 (PE8 個)	1 枚 (PE8 個)	0 枚
PCS	64 枚 (PE512 個)	16 枚 (PE128 個)	4 枚 (PE32 個)
SS	2 枚 (PE16 個)	2 枚 (PE16 個)	1 枚 (PE8 個)

これらの基板は SCU のメインボードを除いて Versa Module Europa(VME) のトリプルハイトの筐体に挿入される (図 11 参照)。この筐体の中にはバックプレーンが取りつけられており、各ステージは一部を除いてバックプレーンにより接続される。他にもバックプレーンは、*ReVolver*/C40 に電源やクロック信号、およびリセット信号を供給する役割もはたしている。SCU はホストマシンの PCI スロットに挿入され、*ReVolver*/C40 の各ステージとの通信にはフラット

ケーブルを用いる。

2.5.5 *ReVolver*/C40-demo の構成

ReVolver/C40 では、RCS を用いて視線を生成し PCS を用いてピクセル値を計算し SS を用いてシェーディング等の処理を行いディスプレイに表示しこれらの 3 つの基板を SCU を使いホストから制御している。

しかし、現在確実に動作可能な基板は RCS が一枚、PCS が 4 枚、SS が 1 枚で、プロトタイプ *ReVolver*/C40 を構成することはできない。また、RCS を用いて視線を生成する場合、PCS は 1 枚しか動作させることができないため、このような構成では PCS のメモリの制限により、64^3 のボリュームデータを可視化することしかできず、それより大きなサイズは実行できない。

そこで、RCS のかわりにホストコンピュータで視線を生成し PCS、SS を用いてボリュームレンダリングを実行する構成をとっている。

第3章　ReVolver/C40の基本性能評価とそれに基づく高精細化の検討

この章では、*ReVolver*/C40の基本性能評価を行い、*ReVolver*/C40のデータサイズに対するスケーラビリティを示す。また新システムに向け、*ReVolver*/C40の画質による評価と、更なる高速化のための様々なピクセル値計算の高速化手法の評価も行う。

3.1　基本性能評価

これまで、RCS,PCS,SSを用いた*ReVolver*/C40ではRCSのハードウェアの制限から、PCSは8PEまでしか動作させることができなかった。そのため、PCSにおいてPE数とデータサイズを増やした場合の性能評価をすることはできなかった。そのため、RCSを用いることなく画像生成を行える*ReVolver*/C40-demoを用いることでPCSのPE数を32台とし、PE数とデータサイズに関する評価を行う。

3.1.1　SCUの評価

ReVolver/C40では、RCSにおいて視線を生成し次のステージへ転送していたが、*ReVolver*/C40-demoではそれをSCUとホストで行っている。それらの速度と実測により求め、*ReVolver*/C40と比較する。また、可視化対象のデータサイズが大きくなれば転送速度の重要性も増すため、SCUを介したデータ転送速度も実測により求める。

ホストからSCUを介したPCSへのデータ転送速度　ホストからSCUへのデータ転送には、使用したDSPの専用通信ポート (以下 Com Portと呼ぶ) を用いる。このデータの転送速度を、ホストが送信したデータをPCSが受信するまでの時間として、PCSが受信終了したらSSが画像を表示するようにし、その時間を実測したところ、$4.04MB/sec$の性能であることがわかった。本来、Com Portは$20MB/sec$のはずであり、PCIバスの転送速度も$133MB/sec$のはずで、Com Portの転送速度である$20MB/sec$のはずだが、実際はその約5分の1の性能となった。これには、ホストPC側のメモリアクセス時間とPCSのDSP側のメモリアクセス時間も入るためである。

ボリュームデータ転送速度　*ReVolver*/C40では対象となるボリュームデータをPCSのメモリに格納する必要がある。また、途中でボリュームデータを変

更するといった要求にも答えなければならない。ボリュームデータの転送時間を実測したところ、256^3 分のデータ $(256^3 \times 1byte = 16MB)$ の転送に 19.8 秒かった。前述のように *ReVolver*/C40 ではボリュームデータを三重化して保持するため、受けとったデータのアドレス計算とメモリアクセスが必要なためである。

視線生成速度　本来 *ReVolver*/C40 では RCS で実行されるはずであった視線の生成処理を、PCI バスに SCU ボードを挿してあるホストマシンで行わなければならない。そのホストマシンでの視線の生成時間を実測したところ、256^2 のスクリーン分の視線の生成には、106.1 ミリ秒かかることが分かった。RCS を用いた場合、256^2 のスクリーン分の視線の生成には 115.73 ミリ秒かかることがわかっているのが、ホストで計算を行った場合は 106.1 ミリ秒とほぼ同じ速度で *ReVolver*/C40-demo でも処理できていることがわかる。

3.1.2　PCS の評価

データ転送速度　*ReVolver*/C40-demo では PCS 間の接続は Com Port を用いている。このデータ転送速度を、PCS で送信側 DSP が送信開始してから受信側 DSP が受信終了するまでの時間として、データ転送速度を実測したところ、約 $4.4MB/sec$ の性能となった。この結果も Com Port の転送速度を大きく下回っているが、これは DSP のメモリアクセス時間も含めて実測しているからと考えられる。また、同じ SCU でありながら結果が異なるのは、ホストである PC と DSP のメモリアクセス時間の違いと考えられる。

画像生成速度　*ReVolver*/C40-demo での 32PE での画像生成速度を実測した。256^3 のボリュームデータを 256^2 のスクリーンに描画する際の計算時間は、一枚 2.60 秒かかり約 $0.38fps$ となった。128^3 のデータを 128^2 のスクリーンに描画するには 0.40 秒で約 $2.50fps$、64^3 のデータを 64^2 のスクリーンに描画するには 0.06 秒で約 $16.7fps$ となった。また、256^2 のスクリーンに描画する際に、各 PE の担当するスライスの数と実測による描画速度を表 2 にまとめる。

このことから、ボリュームレンダリング実行時の 1 スライス $256^2 \times 1$ のボリュームデータ当りの PCS の計算時間は 0.3 秒、通信時間は 0.2 秒であることがわかる。

3.1.3　考察

ReVolver/C40-demo では、画像生成は *ReVolver*/C40 と同じく PCS によって律速されており、視線の生成処理は *ReVolver*/C40 で RCS を用いた場合とほぼ

表2: PE当りのスライス数と画像生成速度

スライス数	1	2	3	4	5	6	7	8
時間(秒)	0.50	0.80	1.10	1.40	1.70	2.00	2.30	2.60
fps	2.00	1.25	0.91	0.71	0.58	0.50	0.43	0.38

同じ速度で終了し、ホストで行っても問題ないといえ、*ReVolver*/C40-demoを*ReVolver*/C40のアーキテクチャを利用した32PEのシステムとして評価してよい。

また、実際に使用するには*ReVolver*/C40でも*ReVolver*/C40-demoでもホストPCからPCSへのボリュームレンダリングの転送が必要で、その時間は画像生成に必要な時間に較べて非常に大きい。同一のボリュームデータに対して何度のレンダリングを実行するような場合はこのデータ転送時間は問題とはならないが、頻繁にボリュームデータを変更し、レンダリングを実行するような場合はこの時間がボトルネックになるものと考えられる。この時間を減らすためには、ボリュームデータの並列転送が解決法として考えられる。

さらに、*ReVolver*/C40ではディスプレイへの出力を前提としているため、実測にはこのディスプレイの変化で行う他なく、かなりの誤差が生じてしまう部分もあった。

3.2 生成画像の高画質化と描画速度のトレードオフに関する検討

平行投影ではスクリーン上の全ての視線が同一であるが透視投影では全ての視線が異りピクセルの数だけ計算量が増加する。また、平行投影では無限直方体であるビューボリューム(図17参照)[4]が透視投影では無限ピラミッド型であるため、視点から奥へ行くに従って広がる。この広がりを考慮して計算しなければならないため、この点でも平行投影に較べ透視投影は計算量が増加する。

ReVolver/C40では、レイキャスティング法により3次元空間を2次元のスクリーンに投影しており、その際には平行投影と透視投影の両者を用いて行えるが、透視投影を用いて画像生成を行う際のビューボリュームの広がりが生成画像に与える影響は少ないものとして、平行投影と同じようにビューボリュームを直方体として扱い広がりを無視することで、高速化を実現している。

また、*ReVolver*/C40では、サンプリング点のボクセル値を連続モデルと離散

モデルの両方で扱い画像を生成できる。両者の実行時間には大きな差があるが、生成画像の画質にも大きな差があることが考えられる。

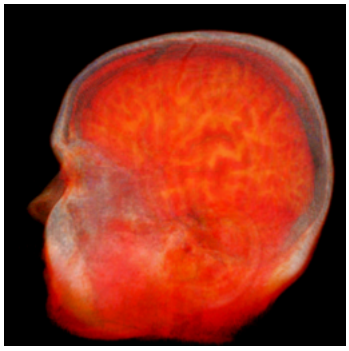

図12: サンプルデータ

これらの *ReVolver*/C40 における高速化手法が画質に与える影響を評価するため、サンプリング方式に関して、1) 連続モデルと離散モデル、2) ビューボリュームの簡略化による高速化の観点から、生成される画像の画質とそれに要する時間を評価しその必要性を考察した [5]。この評価には人間頭部のボリュームデータ (ボクセル 256^3 個) を用いて計算を行い得られた画像 (図 12) を用いた。

また、これらの評価は *ReVolver*/C40 上で行うことはできるが、*ReVolver*/C40 はディスプレイへ出力を前提にしているので統計解析が出来ず、生成画像の主観的評価だけでなく、数値データに基づく統計的解析を行うため、ワークステーション上でソフトウェアでボリュームレンダリングを実行しその結果を用いて評価を行った。

3.2.1 連続モデルと離散モデル

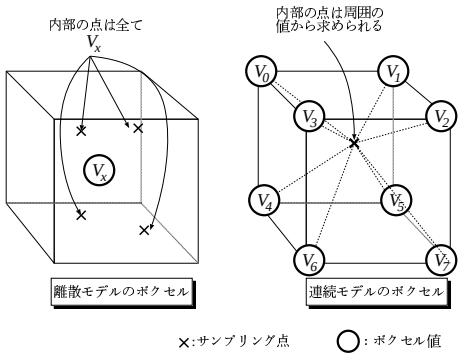

図13: 離散モデルと連続モデル

ボリュームデータは本来連続な3次元データをサンプリングすることにより離散的にボクセルの集合として表現したもので、ボリュームレンダリングを実行する際に、ボクセルを連続モデルとして扱うか離散モデルとして扱うかで画質に大きな差が生じることが考えられる。

連続モデル(図13右)はサンプリング点の周辺格子座標8点のボクセル値からTri-linear補間によりボクセル値を求めるものであり、離散モデル(図13左)はサンプリング点にもっとも近い格子座標上のボクセル値をサンプリング点のボクセル値とみなすもの[2]で、メモリアクセスに関しては、連続モデルは離散モデルがサンプリング点のボクセルへのメモリアクセスのみのところを、周8点のボクセルへのアクセスが必要となる。また、演算量に関しては、連続モデルは離散モデルでは必要のない補完のための計算が必要となる。このように、どちらを採用するかで計算量もまた大きく変化する。この計算量の増加による画像生成に必要となる時間の増加と画質の向上のトレードオフを評価するため、連続モデルと離散モデルの両者を用いて生成した画像の画質を比較する。画像生成は透視投影を用いて行っているが、*ReVolver*/C40はサンプリング法として主軸等間隔サンプリング法をとっているため、平行投影を行った場合はサンプリング点がボクセルの中心点となってしまうため、連続モデルと離散モデルの差がでない。

比較には256^3のボリュームデータ(図12)を256^2のスクリーンに投影することで得られた画像を用いている。視点とスクリーンの位置は、一般的な使用と考えられる、ボリュームデータの中心をスクリーンの中心として、画角60°(図14)のものを標準とし、更に画角を16°(図15)とすることで、対象を拡大して表示したもの、また、ボクセルの境目では連続モデルと離散モデルによる差が大きくなると考え、中心から周辺へとずらすことで、境目を拡大して表示したもの(図16)、の3つを用いて連続モデル・離散モデルの両者を用いて画像を生成した。

画質の比較は、両者の生成画像の各ピクセルごとの輝度の差の最大値を求め、それを両者のピクセルの輝度の最大値の何パーセントに当たるかを計算することで行った。これは、生成画像の輝度に差があれば見た目に違いが生じるはずで、また生成画像の最大輝度が小さければ、輝度の差が小さくても見た目に目

立ちつと考え、このような方法をとった。[1)]

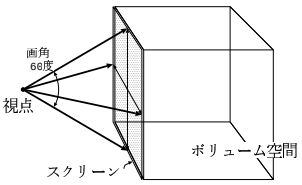

図14: 画角60度

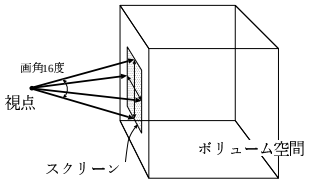

図15: 画角16度(中心)

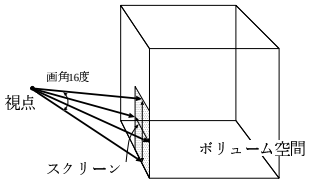

図16: 画角16度(周辺)

生成画像を比較すると、画角60度の場合、両モデル間での対応する連続モデルのピクセルの輝度の差は最大で7.58%、画角を約16度として解像度を4倍にそのままの視点で中心部分を拡大した場合は輝度の差は最大で6.70%、ボクセルの境目であるスクリーンの周辺部分を拡大した場合は最大で8.79%となっている。また、横からだけでなく画角は同じで視点を移動させて下から見た場合18.4%、中心を拡大した場合18.7%、周辺を拡大した場合14.7%となっている。また、実行時間の差は連続が離散の約4倍かかった。

表3: ボクセル値計算方式の違いが画質と計算時間に与える影響

画角	スクリーン	輝度の差	計算時間
60°	中心	7.58%	3.87
16°	中心	6.70%	4.02
	周辺	8.79%	3.99

さらに視点とスクリーンの位置や対象とするボリュームデータを様々に変化させたところ、ボリュームデータが存在する領域全般について連続モデルと離散モデルの輝度値には差があった。また、透明度が低く不透明なボリュームデータの場合、連続モデルと離散モデルで全く異なる生成画像が得られる。

結果として、確かに連続モデルと離散モデルというサンプリング方法の違いによる画質の差は認められ、また、視点とスクリーンの位置、ボリュームデータの透明度という条件によって大きく異なる結果がでることがわかった。従って、実行時間の差も考えると、通常では連続で高速性が求められる場合のみ離

[1)] 輝度の差の最大値は、$\frac{\max|v_{a_{i,j}}-v_{b_{i,j}}|}{\max(v_{a_{i,j}},v_{b_{i,j}})}$　$v_{a_{i,j}}, v_{b_{i,j}}(0\leq i,j,\leq 255)$ となる。

散といったようなに、場合を分けて使用するサンプリング法を変更するという方法がよいだろう。

3.2.2 透視投影におけるビューボリューム簡略化の画質および描画速度への影響

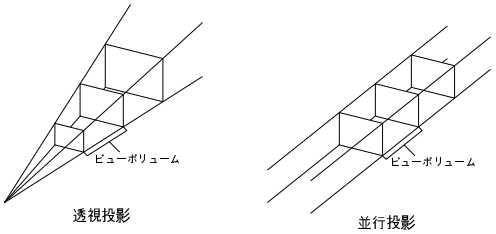

図17: ビューボリューム

ReVolver/C40ではレイキャスティング法を用いてスクリーンに写る部分のボリューム空間を対象として計算を行う。平行投影では、最終的に表示される空間であるビューボリュームは半無限直方体だが、透視投影では、ビューボリュームは、視点とスクリーンの変から構成される半無限ピラミッドとなる。

ボリュームレンダリングでは、このビューボリュームに含まれるボリューム空間内の全てのボクセルを用いて画像を生成しなければならない。平行投影では、ビューボリュームが直方体であるため、視点から奥へ行くについれて広がっていくことがなく、スクリーン上の各ピクセルに対して一本の視線を伸ばしその視線上のボクセルを用いて計算を行い画像を生成すればよい。

ReVolver/C40の透視投影は標準ではビューボリュームの中心線である視線上のボクセルをサンプリングすることで高速化を実現している。すなわち視線の広がりは荒涼するものの、ビューボリュームの広がりを考慮せず平行投影と同じく直方体として捉えることで処理を簡略化している。実際には、そこで、これらの処理の簡略化がどの程度生成画像の画質に影響を及ぼすか、実際にワークステーション上でプログラムを実装しこの処理を簡略化することなく画像を生成し簡略化して生成した画像と比較することで評価した。

このプログラムでは、ビューボリュームの広がりを簡略化することなく処理するために、各視線に隣の視線との距離の情報を持たせ、その距離がある一定値を越えた時点でその視線を分裂させ計算を行う。

具体的には図18のように、視線間の幅がボクセル2個分を越えた場合に視線

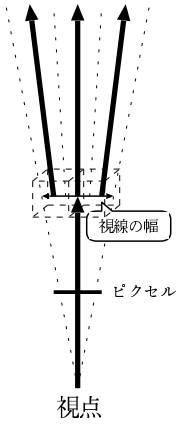

図 18: 視線の分裂

を 3 本に分裂させ、分裂した視線に対する寄与を重み付き加算すること [6][7] でピクセル値を計算する。現在重みは全て 1 として 3 本の視線のピクセル値の平均を元の一本の視線のピクセル値として計算している。視線の幅がボクセル 2 個分となった時点で視線の分割を行う。

計算に関与するボクセルの数を比較すると、ボリューム空間(ボクセル数1677万 (256^3)) の端にスクリーンを配置して画角 60 度で見た場合、ビューボリュームには含まれているが、計算には関与しなかったボクセル数は視線分裂しない場合は 896 万であったのが視線分裂させることにより 410 万に減少した。

生成画像の輝度について比較すると、分裂する/しない両者のピクセルの輝度の差は最大で 1.99% しかなかった。実行時間は分裂が起こる数により大きく異なるが、この場合は分裂する場合の方が約 3 倍の時間を要した。画角を小さくするとビューボリュームの広がりも小さくなるので分裂が起らず、計算時間の差がほとんどなくなった。

表 4: ビューボリュームが画質と計算時間に与える影響

画角	スクリーン	輝度の差	計算時間
60°	中心	1.99%	3.02
16°	中心	0.00%	1.00
	周辺	0.00%	1.02

限られた条件での比較しか行っていないが、ビューボリュームを簡略化するこ

とで高速化を行っても、簡略化しない場合とはそれほどの違いは見られず、簡略化をしないことによる速度低下に比べてあまり画質の向上がみられない。しかし、視点から見てボリューム空間の奥へ行くほど分裂の必要性があるので、有意なデータが奥のボクセルに存在する透明度が高いボリュームデータなどの場合は、分裂することによりもう少し差がでるのではないと考える。

また、画角を広げれば分裂の回数も増加するため、輝度の差がより大きくなることが考えられるが、画角を広げれば対象となる物体の見え方は大きく歪みまたスクリーンに写る物体の大きさは小さくなってしまう。通常の使用においては、視線分割は不要であると考える。

3.2.3 まとめ

離散モデルと連続モデルによる生成画像の差は画像全体に渡って見られ、さらにボクセルが存在する部分としない部分との境界や、ある部分を拡大した場合などで大きく見られる。速度の差は離散の方が約4倍高速となったが、生成画像の差はこの速度低下を招いたとしても連続モデルで計算を行わなければならない程大きいものだといえる。

視線を分裂させることに関しては、そもそも分裂しなければならない場合があまりなく、更に起きたとしても視点から見たボリューム空間の奥の方なので生成された画像ではその差は見られない。また、視点から見てボリューム空間の最奥の一面に格子状のデータを配置し、それを視線を分裂させる方法により得られる生成画像では、差があるという結果も出ている [7] が、このような特殊ケースでは視点位置を工夫することで、十分に対応可能であると考える。

今回の評価では速度を犠牲にできる程の画質の向上は見られなかったため、*ReVolver*/C40 によるボリュームレンダリングでは、ビューボリュームが四角錐で広がっていくということは考慮する必要はなく通常は簡略化した表示で十分といえる。

3.3 ReVolver/C40 考察

ReVolver/C40 を用いたボリュームレンダリングで、画像生成速度を支配しているのはピクセル値の計算で、この部分の高速化が更なる課題となる。*ReVolver*/C40 は、ボリュームレンダリングへのアクセスへの局所性を考慮することなく、あらゆる視点に対応するアーキテクチャであるが、ハードウェアを実装し、ソフトウェアで実装してみて、ボリュームデータへのアクセスには、一

回の画像生成を単位として考えると、時間的な局所性はそれほどないが空間的局所性はかなりあるということがわかった。

ReVolver/C40 は主軸等間隔法でサンプリングを行うので各 PE がアクセスするスライスはスクリーンサイズ分の計算が終わるまでは同じとなる。しかし、スキャンライン単位で視線を生成していたのでは、図 19 左のように連続アクセスにはならない。しかし、このアクセスの順序は視線を生成する際に連続となるように変換可能で図 19 右のようにすることができる。このように変換することで、ボリュームデータへのアクセスの空間的局所性を利用し、キャッシュシステムを用いることで更に高速化が可能といえる。

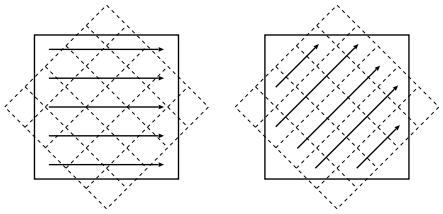

図 19: 連続アクセスへの変換

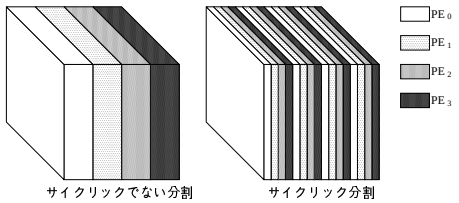

図 20: サイクリック分割

またボリュームレンダリングは *ReVolver*/C40 のようなパイプライン処理との親和性が非常に高く、パイプラインの段数を増やしメモリを増加させることで更に大規模なボリュームデータの可視化へも対応できるといえる。現在 *ReVolver*/C40 では x, y, z 軸に対して分割し順番に各 PE に割り当てている (図 20 左) が、これをサイクリックに割り当てる (図 20 右) ことで、負荷を分散させることが可能で、また、サイクリックに割り当てることで、ピクセル値が不透明となりそれ以後の計算をしなくてよい場合など、その場で計算を終了させるといったことが可能となる。*ReVolver*/C40 の PCS はリング構造をとっていないためサイクリック分割は難しいが、これはハードウェアの改造により対処できる。

このような *ReVolver*/C40 のスケーラビリティとピクセル値計算の高速化手法を利用した、更に大規模なデータに対応する新システムの提案を行う。

第4章　リアルタイム可視化システム

前述のとおり、*ReVolver*/C40 のアーキテクチャは、データサイズに対してスケーラビリティがあるため、このままのアーキテクチャでさらに大きなデータサイズにも対応できる。我々は、このアーキテクチャを利用して更に大規模なデータの可視化に対応する新システムの実現を目指している。

しかし、現在の *ReVolver*/C40 ではデータ転送の問題があり、シミュレーションなどの科学技術計算を行うシステムと連携し、その計算の途中の可視化 [8][9][10][11][12][13][14] などは難しい。新システムでは *ReVolver*/C40 のアーキテクチャを生かしさらに大きなサイズのデータの可視化に対応するとともにこのようなリアルタイム可視化にも対応したい。

本章ではこのようなリアルタイム可視化に対応したシステムの概要を示し、その利用形態、要求仕様、構成方針をまとめ、我々が目指す目標とその構成方針を示す。また、我々の構成方針の実装例を示す。

4.1　システムの概要

これまでのシステムは、シミュレーションの最終結果の可視化、CT スキャンの観測結果の可視化など、最終結果の可視化に主眼が置かれていた。そのため、可視化を行う部分と可視化対象となるデータを生成する部分とにきっちり分けられており、計算サーバなどのデータ生成系で行った計算の最終結果をネットワークなどで、グラフィクスワークステーションやパソコンなどの可視化専用のシステムに転送して可視化するものが一般的であった。

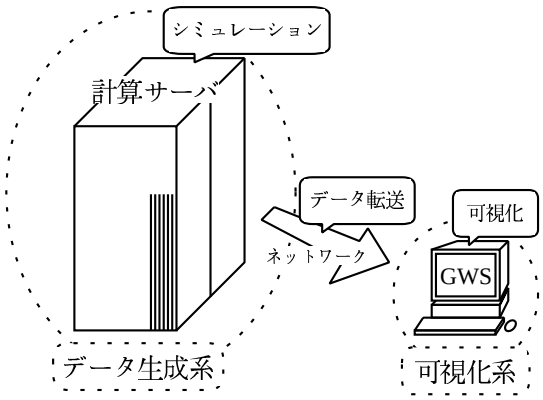

図 21: 従来の可視化システム

しかし、このようなシステムでは計算や観測の最終結果のみで全てを判断しなければならないため効率が悪い。例えば、シミュレーションを例にあげると、初期パラメータの設定ミスなどのような少し計算を進めればすぐ分かるミスも、これまでの可視化システムでは計算終了まで待たなければ分からない。より効率を上げるためには、観測や計算をその時々でリアルタイムに可視化するシステムが必要となる。

また、これまでのシステムでは、データ生成系と可視化系は別のシステムであったため、シミュレーション側のデータ形式と可視化側のデータ形式が異なるといったデータの互換性であったりシステムの操作性など、様々な問題があった。そこで、リアルタイム可視化システムには、データ生成系と可視化系を統一的に扱うインターフェイスも必要となる。

次に、このようなリアルタイム可視化システムの利用形態を示しそれぞれの要求仕様をまとめる。

4.2 システムの利用形態と要求仕様

リアルタイム可視化システムの利用形態を述べ、それぞれに必要となる仕様をまとめる。リアルタイム可視化システムの利用形態としては、これまでのような最終結果を高速に可視化し解析するオフライン可視化と、途中経過をリアルタイムに可視化するオンライン可視化の 2 つが考えられる。

4.2.1 オフライン可視化

これまでの *ReVolver*/C40 がターゲットとしていた分野で、医療画像の解析やシミュレーションの最終結果の解析等の利用形態である。データ生成系から可

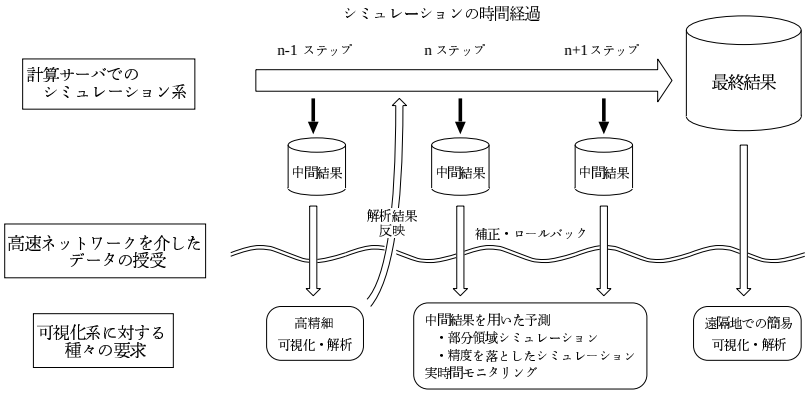

図 22: リアルタイム可視化システム

視化系への可視化対象データなどの受渡しは頻繁でなく、両者が比較的粗に連携している場合である。 オフライン可視化では、対象を何度も可視化して解析することが考えられるので、可視化系にはできるだけ高精細かつ高速な画像生成とインターフェイスの実時間応答性、操作性が求められる。また、あらゆる方向からの視点への対応など、可視化の際のパラメータに対する要求が高い。

4.2.2　オンライン可視化

新しく必要となりつつある利用形態で、シミュレーションの途中経過の可視化に代表される利用形態で、データ生成系から可視化系への可視化対象データの受渡しが頻繁に起こる場合である。そのためオンライン可視化では、できるだけ高速な可視化対象データの受渡しとが求められるが、オフライン可視化のように同じデータを何度も可視化し解析するような利用形態ではないので、画像生成速度に対する要求はあまり高くない。また、視点を移動させながら途中経過の観測などはあまり考えられないので、可視化の際のパラメータに対する要求は低い。当然、データ生成系と可視化系の間の接続には高い性能を求められる。

これらの二つ利用形態に対応するためには、これまでよりも可視化系とデータ生成系の連携が重要となる。そのためシステム構成としては以下の方法が考えられる。

4.3　システムの構成方式

方式 1

まず考えられるのが、これまでの可視化システムと同じように、計算サーバにデータ生成系を置きそれとは別に可視化専用のグラフィクスワークステーションのようなものを設けそこに可視化系を置く方式である。

このような方式では、可視化系を専用に設けるため、オフライン可視化で求められる可視化系の性能を満たすことは容易であるが、オンライン可視化で求められるデータ生成系と可視化系との間の接続に対する性能を満たすことが難しくなる。

方式 2

オフライン可視化で求められるデータ生成系と可視化系との間の接続に対する性能を満たすためには、データ生成系と可視化系を別々に設けるのではなく、例えば同じ汎用の大型計算機内に設けるといった方式も考えられる。

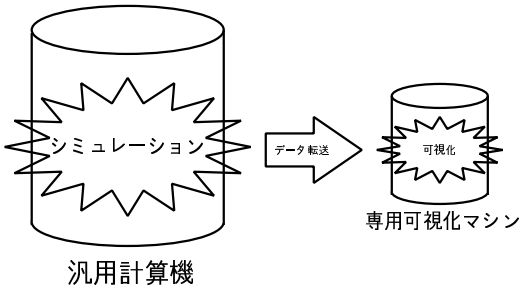

図 23: 方針 1

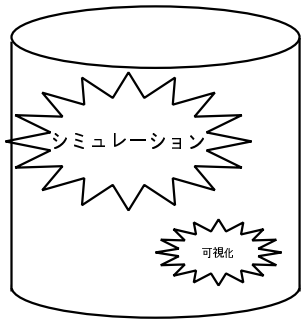

図 24: 方針 2

この方式では、両者の接続に対する性能は容易に満たすことができるが、可視化に特化したシステムではないため、可視化系の性能を満たすことが問題となる。

4.4 新システムの構想

以下に、我々が提案するリアルタイム可視化システムを実現のための新システムについて述べる。

まず我々の目標を示し、その実現のための要求仕様をまとめる。その要求仕様を満たす構成方針を示し、またその実現可能性を示すために実装例を上げる。

我々の目標とする可視化対象データは、X^3 のシミュレーション結果から得られる N^3 のボリュームデータで、 その可視化結果を S^2 のスクリーンに表示する場合を考える。N の値としては 4096、 S の値としては 2048 [1)] を当面の目標とする。また、ボリュームデータはボクセルの種類は 256 種類で、それぞれ色・透明度を $4byte$ で保持するものとする。

4.4.1 要求仕様

ボリュームレンダリングに関する要求は以下の通り。4096^3 のボリュームデータは、ボクセルが 256 種類なので $1byte$ で保持でき、$N^3 = 4096^3 \times 1byte = 64GB$ の容量となる。このボリュームデータを $30fps$ で 2048^2 のスクリーンに表示することを考えると、計算が簡単な並行投影モデルでもメモリバンド幅は $S^2 \times N \times 30 = 2048^2 \times 4096 \times 30 = 480GB/sec$ 必要となる。演算性能に関しては、ボクセル一つ当り透明度の計算で 1 回、RGB それぞれの色の計算で 4×3 回で、合計 13 回の浮動小数点演算が必要となる。従って、$2048^2 \times 4096 \times 30times13 = 6.5TFLOPS$

[1)] Super High Definition 規格相当

の演算性能が必要となる。

表5: システムの要求仕様

メモリ容量	$64GB$
メモリバンド幅	$480GB/sec$
演算性能	$6.5TFLOPS$

4.4.2 新システムの構成方針

システムの構成方針としては、4.3で述べたのように可視化系とデータ生成系を分ける方法(方針1)と一体とする方法(方針2)がある。

そこで我々の提案する方式は、方針2のようにデータ生成系と可視化系を一体化するというアプローチをとりつつ、可視化のための専用システムを実現する。具体的にはPCクラスタをデータ生成系として用い、可視化処理をPCクラスタを用いてソフトウェアで行うのではなくAGPあるいはPCIスロットへのアドイン型の専用可視化ハードウェアで実現することで、ボリュームデータの高速かつ並列転送を可能としかつ目標とする高速な可視化と操作性を保証することができると考える。

以下に、我々の構想する新システムの構成方針を、ボリュームデータの転送速度と描画速度、描画方針、ユーザインターフェイスの観点でまとめる。

転送速度と描画速度

オンライン可視化を考えた場合、データ生成系から可視化系へのボリュームデータの転送が頻繁に行われるため、ボリュームデータのサイズが小さい場合はあまり問題とならないが、我々のシステムが目標とするような大規模なデータとなると、ボリュームデータの転送方法が大きな問題となる。

ボリュームデータの転送時間について考えると、方式1のようにデータ生成系と可視化系が独立したシステムの場合、両者の間のネットワークを強化しボリュームデータを圧縮して送るといった、極めて直接的な解決策をとらなければならない。

このような方式を採用している実装例として、PAVEMENT[9]があげられる。PAVEMENTは、シミュレーションを実行する超並列計算機と可視化を実行する専用のグラフィクスワークステーションから構成されるシステムで、従来は

シミュレーションの最終結果をネットワークを用いて転送し可視化を行っていたものを、リアルタイム可視化に対応させるため両者をつなぐネットワークの物理的な数を増やし並列に転送することで高速にしたものである。

一方、方式2のようにデータ生成と可視化処理を一体化して同一のシステム内で可視化処理をソフトウェアで実現すると、ボリュームデータの転送自体をなくすことができ、データ転送の問題は解決される。

このような方式を採用している実装例として、RVSLIB[11]があげられる。RVSLIBでは、可視化の対象となるシミュレーションの計算量と比較して可視化のために必要な計算量は無視できる程であるので、汎用の大型計算機内でシミュレーションと可視化の両者を行いデータ転送を全く無くし、リアルタイム可視化に対応している。この方式は、データ転送が全くないため確かにオンライン可視化には最適といえる。ボリュームデータの転送に関しては、RVSLIBのような方式2をとるべきといえる。次に描画速度に関して考える。

オフライン可視化時には我々の新システムでは、4096^3のデータを2048^2のスクリーンに$30fps$で表示することを目標としている。ボリュームレンダリングでは、1つのボリュームデータ当りの演算量は比較的少なく、描画速度を支配するのはボリュームデータへのアクセスバンド幅と考えて良い。新システムで必要となるメモリバンド幅は$480GB/sec$となっている。

ボリュームデータの転送に関して考えた場合、方式2のようにデータ生成系と可視化系を一体とするアプローチをとるべきである。しかし、オンライン可視化時にも目標とするの描画速度を実現すると、シミュレーションにも負荷をかけてしまう。従って、シミュレーションに負荷をかけることなく目標とする描画速度を得るためには、可視化のための専用のシステムが必要となる。

ボリュームレンダリング

我々の提案するボリュームレンダリングの実行方式は、既に開発した$ReVolver/$C40で採用したアーキテクチャをベースにした物で、$N \times N \times L$[1)]のサブボリューム単位で並列化し、1ピクセル分のピクセル値計算をサブボリューム単位でパイプライン処理することで目標とするの描画速度を得る。

さらに、1)ボリュームデータの圧縮や2)不可視ボクセルのピクセル値計算の省略により、ピクセル値計算パイプラインの負荷を軽減する。またこの高速化

[1)] 並列度をPとするとL=N/P

によって生じるピクセル値計算負荷の不均衡を軽減するために、隣接ノード間で負荷分散を行うことで更なる高速化を図る。

また、我々のシステムは対象ユーザとしては研究室のような単位を考えており、このようなユーザが、汎用の大型計算機を占有することは難しい。そこで我々はシミュレーションを行う部分をPCクラスタとし、そのPCIバスにボリュームレンダリング専用のハードウェアを付加した構成をとる。

このようなPCクラスタに専用ハードウェアを付加する方式を採用している実装例として、VGCluster[10]があげられる。ボリュームレンダリングの並列化の手法としては、*ReVolver*/C40のようなパイプラインでボクセル並列処理(図25右参照)を行うものとは異なり、一つの視線をボリューム並列処理する方式(図25左参照)があり、VGClusterではこの手法をとっている。

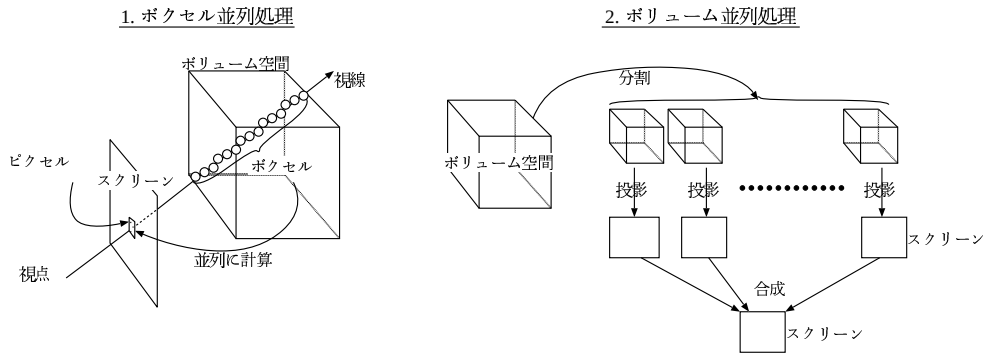

図25: 並列ボリュームレンダイング

VGClusterでは、PCIカード型のボリュームレンダリング専用ハードウェアであるVolumePRO[15]を搭載したPCでPCクラスタを構成し、PCクラスタ部でシミュレーションを実行し可視化をVolumePROを用いて行う。VolumePROは512^3のデータを$30fps$でボリュームレンダリングを実行できるハードウェアで、VGClusterではさらに大規模なデータの可視化を、各スクリーンに対する処理をボクセル512^3ごとに分割し各VolumePROに割り当て最終結果を1つのスクリーンに重ね合わせて表示することで実現している。VGClusterでは、この重ね合わせために専用のハードウェアが必要となっている。

視線を並列に処理する方式では、我々の方式ではサイクリックにパイプラインを割り当てることで容易に実現できる負荷分散や不透明になったら計算を中断して次の視線の計算を開始するといった高速化が難しい。また、データサイ

ズに対するスケーラビリティの面でも同一のシステムを増加させるのみでよい我々の方式が有利といえる。

ユーザインタフェース

従来のオフライン型の可視化システムからの継続性を重視し、ユーザインタフェースとしては汎用可視化ソフトウエアAVSを利用し、AVSの組み込みモジュールを介して専用可視化ハードウェアとのインタフェースを行う。この際、シミュレーション結果からボリュームデータへのマッピング処理が必要となるが、この処理自体も極めて高いメモリバンド幅を要求する処理である。そこで、規則的なマッピング関数が与えられる場合に、マッピング処理をハードウェアで行うアクセレレータを設けることで一連の作業の高速化を図る。

シミュレーションの途中結果の実時間可視化が可能になることで、次なるステップとして、シミュレーション自体の制御を行うメカニズムを提供する。ここでは、シミュレーションの一時停止、中断やパラメータの変更といった操作を実現する。

4.4.3 新システムの実装可能性

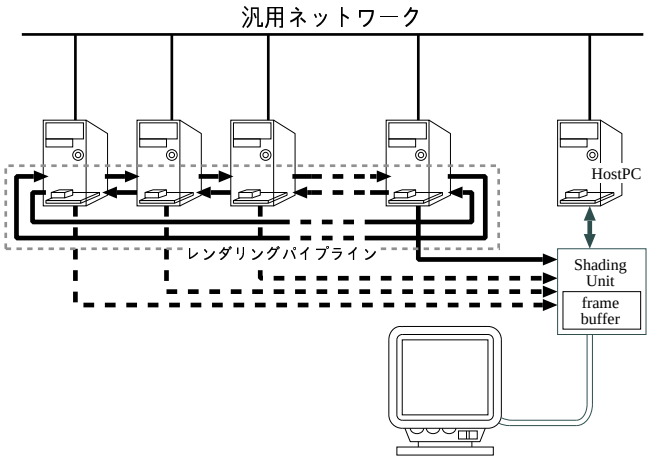

図26: 実時間可視化システムの構成

4096^3 の 8bit ボリュームデータを、SHD規格相当のスクリーン (2048^2) に秒間30枚のフレームレートで出力することを目標とした、リアルタイム可視化システムの構成例を図26に示す。図は、128ノード構成のPCクラスタに、ボリュームレンダリング向けアクセラレータ(VisA:**Vis**ualization **A**ccelerator、図27参照)を装備し、VisA間を双方向高速リンクで接続する。計算結果はVisA

間リンクと同一規格のケーブルによりフレームバッファへ送られディスプレイへ表示される。生成された2次元画像に対して、さらに高度な後処理が必要な場合は、ピクセル毎のZ値や方向等の付加的な情報とともにホストPCに送り、画像のアーカイブや汎用のグラフィックスアクセラレータを利用した表示を行う。

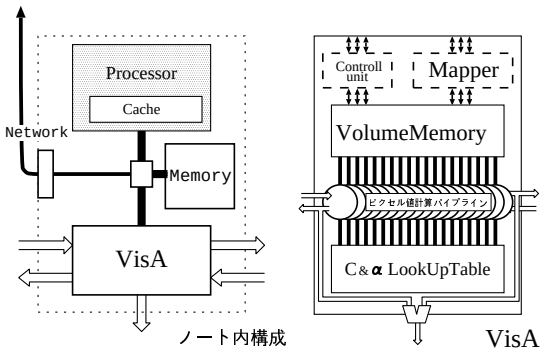

図27: 内部構成

ReVolver/C40では、視線生成、ピクセル値計算、シェーディングの3ステージをそれぞれ専用のハードウェアで構成したが、今回提案するシステムではピクセル値計算のみを重点的に専用ハードウェア化し、その他のステージで必要であった処理は各PCのCPUや汎用のグラフィックスカードを用いて実行する。以下、主なシステム構成要素について説明を行うとともに実装可能性について検討する。

PCクラスタ部

128ノードのPCクラスタ部は、視線生成処理を行うとともに、オンライン可視化時にはシミュレーションを実行する。

VisA間リンク

このリンクは、レンダリングパイプラインにおいて計算途中の色情報、透明度情報、Z値を送るためのリンクであり、$1008MB/s$の転送速度が必要である[1)]。各ノードに隣接ノード間の双方向リンクと、フレームバッファへの出力ポートを設けることで、*ReVolver*/C40で対応が困難であった種々のレンダリングアルゴリズム[1][6]に対応する。具体的には、DVI用LVDSインタフェースを用いてネットワークを構成することを検討している。

1) フレームバッファの入力側スループットの約二倍

ボリュームメモリ

ボリュームデータを格納するための容量2GB のメモリで $4000^2 \times 32$ のサブボリュームを4 セットまで格納可能とする。 ノード内のピクセル値計算ユニットに対して4GB/s のバンド幅を確保するため、PC800 規格相当のRambus chanel 4 チャネル (合計 6.4GB/s) で構成する。

ピクセル値計算パイプライン

32 個のピクセル値計算ユニットをパイプライン接続して構成する。最も単純なサンプリングアルゴリズムを採用した場合、 1 つのボリュームデータに対して、 RGB それぞれに、乗算2 回+加 (減) 算2 回、 透明度に対して乗算1 回の演算が必要である。 表示に必要なRGB は8bit であるが、誤差伝搬の影響を軽減するため16bit 固定小数点として色と透明度の演算を行う。 パイプライン周波数は、画面サイズとフレームレートから128MHz となる。

C&α LUT

RGB 各8bit の色情報と、8bit の透過率を保持する256 エントリのルックアップテーブルである。 各ピクセル値計算ユニットが1 ボクセルの演算を行う度に1 回参照されるため、スループット的にはボリュームメモリの 4 倍の性能が要求されるが、小容量のメモリであるためマルチポート RAM を複数個用いて実装可能と考えられる。

制御ユニットおよびMapper

制御ユニットは視線情報を始めとする描画に関する情報をCPU から受け取り、VisA 全体の制御を行う。 Mapper はCPU 側からボリュームデータを受け取り、ボリュームメモリに格納する。 アクティブレンダリングを行う場合において、シミュレーション結果からボリュームデータへのマッピング処理が定型的かつ簡易なものであれば、 CPU 側でのマッピング処理を省略し Mapper に直接マッピング処理を行わせることで高速化を図ることができる。 この目的のためにMapper にはFPGA 的な機能を持たせる。

4.4.4 まとめ

この章では、我々が目指すリアルタイム可視化システムについて述べ、*ReVolver*/C40 のアーキテクチャを利用して 4096^3 のデータを 2096^2 に $30fps$ でボリュームレンダリングを用いて描画する新システムを提案した。次章では、新システムの実現に向けた様々なテスト環境として *ReVolver*/C40 を用いて実装したプロトタイプシステムについて述べる。

第5章　ReVolver/C40でのリアルタイム可視化

我々の目標とする新システムは、シミュレーション等の科学技術計算を行なうPCクラスタに、可視化専用のハードウェアをPCIバスあるいはAGPバスに付加することで実現する。この新システムを実装する前段階として、新システムのための様々なテストを行うために、*ReVolver*/C40を用いてリアルタイム可視化システムを実装した。

ReVolver/C40では、ボリュームデータの転送を単一のネットワークで行っているため非常に時間がかかる。そのため、リアルタイム可視化を行う際のボトルネックとなる。このボトルネックを解消することを目指し、プロトタイプであるこのシステムでも並列なデータ転送を行えるようにした。また、新システムでもデータ転送を並列に実行するため、新システムのデータの並列転送とシミュレーションとの関係などのテストを行うこともできる。

5.1　プロトタイプシステムの構成

プロトタイプシステムは、シミュレーション等の計算を行うPCクラスタと、そのPCIバスに付加する通信ボードと*ReVolver*/C40のPCSとで構成される。プロトタイプシステムは、新システムのように、PCIバスに可視化ハードを付加するのではなく、*ReVolver*/C40のピクセル値計算を行うPCSを用いて実現しており、通信ボードはPCクラスタとPCSとのデータの送受信のためのボードとなっている。しかし、*ReVolver*/C40は小さなサイズのデータに対する可視化なら十分な性能を持っており、通信ボードによるデータ転送の高速化が実現できれば、十分リアルタイム可視化システムのテスト環境となりえると考える。

5.1.1　PCクラスタの構成

PCクラスタは、Intel PentiumIII 1GHz 512MBのパソコンを五台、OSはVine Linux 2.1.5 (kernel 2.2.18)で構成している。ネットワークには100BASE-TXのイーサネットを用いており、並列化はMPIを用いて処理している。

5.1.2　通信ボードの構成

通信ボードは、シミュレーションのデータをPCクラスタからPCSへ転送するモードと可視化処理実行時にPCS間の通信を行う二つのモードの制御を行う。

通信ボードの構成要素は、以下の二つである。

メインボード　メインボードは、PCSと接続するCom Portのコネクタと、Xil-

inx 社の Field Programmable Gate Array:FPGA(XC4062XLA) で構成されており、各PC クラスタのPCI スロットに接続する。

サブボード　サブボードは、メインボードからフラットケーブルでもたらされる信号を、二つの Com Port に分岐する基板である。

5.1.3　全体の構成

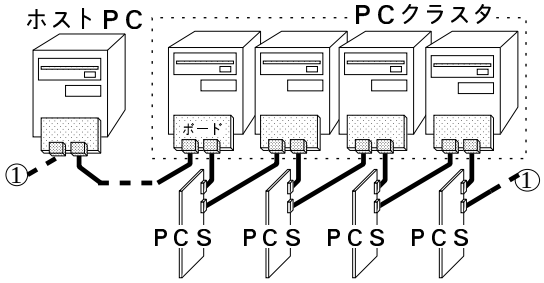

図28: プロトタイプシステムの構成

全体の構成は図28 のように、PCI バスに通信ボードが挿さったパソコンが五台からなるPC クラスタと四枚のPCS 基板 (合計で32DSP 構成) からなる。四台のPC クラスタでシミュレーションを実行し、五台目のPC クラスタは、システム全体の制御を行う。具体的には、シミュレーションの実行開始・終了、データ転送の開始・終了、ボリュームレンダリングの開始・終了の全体の制御と、ボリュームレンダリング実行時の視線生成・計算結果である生成画像の受信とディスプレイへの表示を行う。

5.2　プロトタイプシステムの実装

5.2.1　アプリケーションプログラムの実装

PC クラスタでの科学技術計算のプログラムとしては、与えられた境界条件の下で熱拡散方程式を解くプログラムを MPI 通信ライブラリを用いて実装した。データの分割は、256^3 の 3 次元データを $256^2 \times 64$ のデータに四分割し、それぞれのPC クラスタに割りあてている。周囲 8 点の温度と自分の温度を含めた平均を次サイクルの自分の温度とし計算を行い、隣接クラスタ間で通信を行うことで熱拡散のシミュレーションを実行する。

5.2.2　通信ボードの実装

通信ボードが転送を行うプログラムおよびデータは以下の通り。

1. PCS の DSP へのプログラムのダウンロード
2. PC クラスタで実行中のシミュレーションデータの PCS への転送
3. ボリュームレンダリングの際の視点やスクリーン位置等 PCS への転送
4. ボリュームレンダリングの実行時の PCS 間通信のバイパス
5. 描画画像の PCS から PC クラスタへの転送

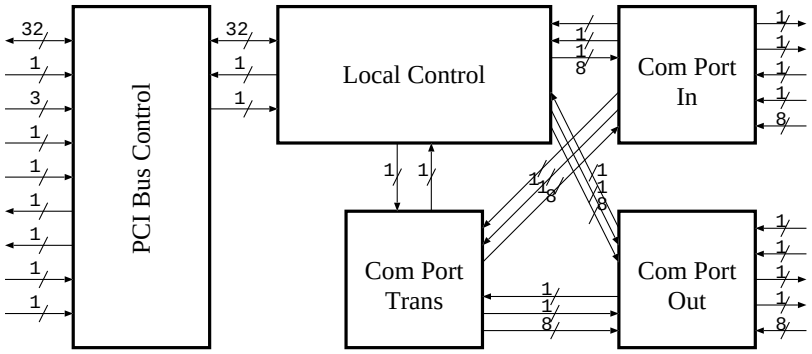

図 29: メインボードの概略

通信ボードには、Com Port を用いた PCS とのデータの送受信と、PCI バスを用いた PC クラスタとの通信のためのインターフェイス回路、PC クラスタと PCS の通信時の制御を行うインターフェイス回路、PCS と PCS 間の通信時の制御を行うインターフェイス回路をハードウェア記述言語を用いて設計した。概略図を図 29 に示す。

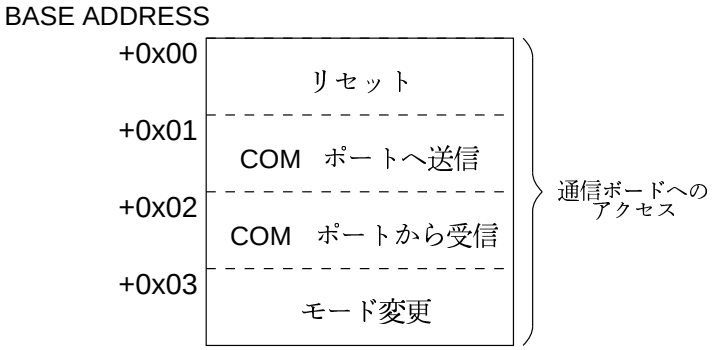

図 30: メモリ空間へのマッピング

- **PCI バスインターフェイス**

  PC クラスタから PCS へのアクセスは、PCI バス上のメモリアクセスで実

現している。Com Port への送信、Com Port からの受信、転送モードの変更を、図30のようにホストマシンのメモリ空間にマッピングし、メモリアクセスにより制御を行う。

- **Com Port**

  Com Port を用いたデータ転送では図31のようにハンドシェイクを行う。PCS 側の DSP が非同期にハンドシェイクを行うのに対して通信ボードは PCI バスの $33MHz$ のクロックに同調してハンドシェイクを行うため、4 バイトのデータの転送を行うのに、16 クロック必要となる (図31参照)。

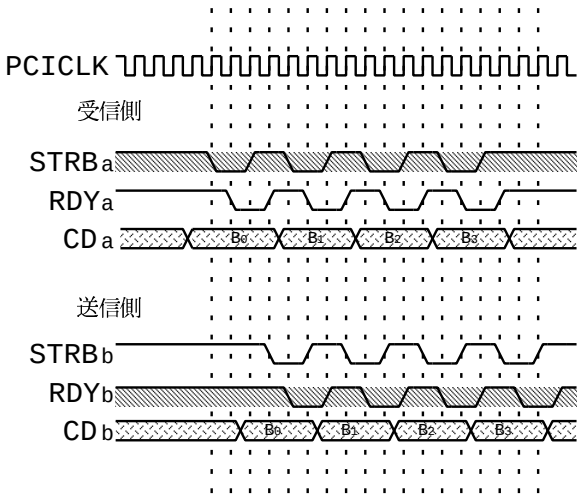

図31: Com Port を用いたデータ転送

5.2.3 全体の実装

ホスト PC での PC クラスタのシミュレーションの制御は以下の順序で行う (図32参照)。

1. 一定間隔PC クラスタがシミュレーションを行ったら自分に接続されている PCS にデータを転送する。
2. 転送が終了したらホスト PC に知らせ、ホスト PC は全PC クラスタの通信ボードを、可視化のためのデータ転送モードにする。
3. 三重化が必要ならば、全ての PCS 間でデータと一巡させる。
4. ホスト PC から視線データを送信し、最終結果をホスト PC が受け取り、自身のディスプレイに表示する。
5. 可視化を続ける場合はそのままくり返し、終了する場合には各通信ボード

をPCSへのデータ転送モードに変更し、各PCクラスタにデータ転送可能であることを知らせ1.に戻る。

現在では、まだシミュレーションへの可視化側からの割り込みに対応していないため、4.で可視化を行っている際にシミュレーションが一定期間計算を行いデータを転送することになっても、待たなければならないシステムとなっている。また、将来的には、ホストPCで可視化を行うタイミングなどのパラメータを指定すれば、それに応じて自動的にデータ転送と表示が行われるようにしなければならない。

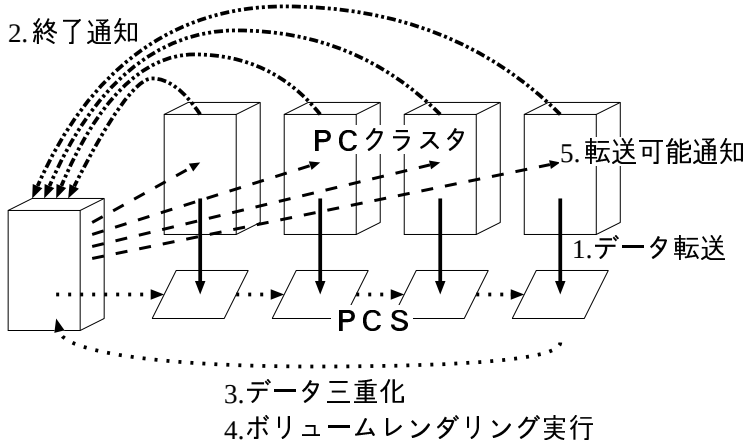

図32: リアルタイム可視化の実行

5.3 プロトタイプシステムの評価

現在の構成では、ボリュームレンダリングを行えるデータサイズが256^3までであるので、四台のPCクラスタそれぞれに、$256^2 \times 64$に分割されたシミュレーションデータを各PCSに転送して画像を生成している。そのPCSへの転送を並列に行う場合と行わない場合を実装し比較を行った。画像生成に際して視点に制限を設けることなくあらゆる方向からの可視化に対応するためには、データを三重化する必要があり、全てのデータを一度全てのPCSを通過させる必要がある。そのために並列に転送を行った場合はその後、PCS内部でデータを巡回させる必要がある。

可視化時間は両者とも同じであるので、可視化までに必要となる時間を実測した。データを並列に転送しない場合、一度PCクラスタ側のネットワークで

データを一箇所に集め、それを通信ボードを用いてPCSに転送することで可視化を行う。このとき可視化までに必要となる時間は以下の通り。

- データをPCクラスタのネットワークによる収集時間4.72秒
- PCSへのデータ転送時間19.8秒
- 合計24.52秒

データを並列に転送する場合に可視化までに必要となる時間は以下の通り。

- PCSへのデータ転送時間4.95秒
- PCS内部でのデータ巡回時間9.9秒
- 合計14.94秒

シミュレーションへの負荷を考えると、シミュレーションが実質停止している時間は、どちらの場合もPCSへのデータ転送時間までで、やはり並列に転送した場合の方が有利であることがわかる。また、PCSへのデータ転送時間がほぼ四分の1に減少したことで、新システムのプロトタイプとして連携ソフトウェアなどのテスト環境に用いる十分な性能を出せるものと考える。

あらゆる方向からの可視化に対応するために三重化を行う場合、内部でデータを一度巡回させる方式もあるが、外部のネットワークでデータを巡回させてそれぞれのPCSに対応する三重化したデータをPCクラスタに分配してからPCSに並列に転送する方式も考えられる。しかし、このような方式では、PCクラスタ側に負荷がかかるため、シミュレーションへの負荷を極力避けるためには望ましくなく、また、データ転送時間の点でもパイプライン転送に対応した内部で巡回させた方が高速に実行できる。しかし、この三重化のための時間はできるだけ減らしたいので、詳細度制御[16]との併用や、可視化できる視線に制限を設け内部のネットワークに空きがある時にバックグラウンドでボリュームデータの三重化のための転送を行うなどの工夫が必要といえる。

また、今回のように*ReVolver*/C40のデータ分割とほぼ同じ分割のシミュレーションばかりではないので、そのようなシミュレーションへの対応も考えなければならない。この場合は三重化しない場合とは異なり、データの再配置なしに可視化を行うことは難しい。そのため、可視化システム内部のネットワークを使用したデータの巡回が必須となる。我々のシステムは、内部のネットワークがリング構造をしているため、このような巡回による再配置は容易に実行できると考えられる。今後はプロトタイプシステムを用いてこのような再配置を必要とするシミュレーションのリアルタイム可視化の評価も行いたい。

第6章　おわりに

我々はこれまでボリュームレンダリング専用並列計算機 *ReVolver*/C40 を開発・実装してきた。*ReVolver*/C40 ではボリュームレンダリングを、視線生成、ピクセル値の計算、シェーディングの 3 つの部分に分けそれぞれ専用基盤を実装し、並列処理を行い高速描画を実現している。実機による評価ではピクセル値の計算が画像生成時間を支配していることがわかった。そこで、このピクセル値計算を高速化するべく様々な高速化手法を実装・評価し、また *ReVolver*/C40 の手法の画質による評価も行った。

さらに、*ReVolver*/C40 はボリュームレンダリングを行うだけのシステムで、可視化による効率の向上はあまり望めない。これから望まれる可視化システムは、科学技術計算などのシミュレーションを実行するデータ生成系とその結果を可視化する可視化系の連携が必須となる。そこで我々は 4096^3 のボリュームデータを 2046^2 のスクリーンに $30fps$ で表示することを目標とする新たな可視化システムを提案した。

新システムはデータ生成系と可視化系を一体とする方式をとるが汎用のシステムでソフトウェアでこの目標を実現するのはメモリバンド幅的に難しいので、可視化には専用のハードウェアを用いることとした。具体的には PC クラスタでシミュレーション等を実行し、PCI バスへ付加するボードで *ReVolver*/C40 のアーキテクチャを応用した可視化専用ハードウェア実装したもので可視化を実行する構成をとる。

また、この新システムの実現には、PCI バス、DVI インターフェイス、PC800 のランバス DRAM などの既存のハードウェアを用いて 128 台程度のシステムで実現可能であることがわかった。

この新システムのテスト環境を構築するためプロトタイプとして、*ReVolver*/C40 の一部である PCS と PC クラスタを接続する PCI 通信ボードを実装し、PC クラスタでシミュレーションを実行し PCS にデータを並列に転送し可視化を行う *ReVolver*/C40 を用いたリアルタイム可視化システムを実装した。結果として、データを並列に転送しない場合の PC クラスタ側のシミュレーションの停止時間は 24.52 秒であるのにに対し、データを並列に転送した場合の停止時間は 4.35 秒で、新システムのテスト環境として十分なリアルタイム可視化システムが実現できた。

今後の課題としては、新システムの実装に向けてさらにシミュレーションと可視化の連携などのソフトウェアのテストをこのプロトタイプシステムで行うこと、さらに、最終的には新システムを実装することがあげられる。

謝辞

本研究の機会を与えてくださった、富田 眞治教授に深く感謝の意を表します。

数々の有用な御指導、御意見を頂いた、森 眞一郎助教授、五島 正裕助手に心から感謝致します。

本研究の共同研究者である原瀬 史靖氏をはじめ、コンピュータ工学講座計算機アーキテクチャ分野の諸氏に感謝致します。

参考文献

[1] 金 喜都他: ピクセル並列処理によるボリューム・レンダリング向きの超高速専用計算機アーキテクチャ, 情報処理学会論文誌, Vol. 38, No. 9, pp. 1668–1680 (1997).

[2] 對馬 雄次他: 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).

[3] 藤代 一成中嶋 正之: コンピュータビジュアリゼーション, 共立出版株式会社 (2000).

[4] 千葉則茂, 土井 章男共著: 3次元ＣＧの基礎と応用, サイエンス社 (1997).

[5] 山内 聡他: 透視投影ボリュームレンダリングにおけるサンプリング方式の評価, 第29回 画像電子学会年次大会予稿集, pp. 33–34 (2001).

[6] et. al, K. L.: An Efficient Method for Volume Rendering using Perspective Projection, *Computer Graphics*, Vol. 24, No. 5, pp. 32–38 (1990).

[7] Kevin Kreeger, e. a.: Adaptive Perspective Ray Casting, *Proc. of 1998 Symp. on Volume Visualization*, pp. 55–62 (1998).

[8] 助村 俊一他: バーチャルマイクロスコープの制御手法の開発, 情報研報 99-HPC-66(HOKKE'97), Vol. 97, No. 37, pp. 25–30 (1997).

[9] 板倉 憲一他: 並列データ流に対する並列可視化, 並列処理シンポジウム JSPP2001 予稿集, pp. 189–196 (2001).

[10] 村木 茂他: VG クラスタ:スケーラブルビジュアルコンピューティングシステム, Vosual Computing グラフィクスと CAD 合同シンポジウム 2001 予稿集, pp. 85–90 (2001).

[11] S.Doi, e. a.: RVSLIB:A Library for Concurrent Network Visualization of Large Scale Unsteady Network Visualization of Large Scale Unsteady SImulation, *SPEEDUP journal 11*, pp. 59–65 (1997).

[12] 小西, 小笠: リアルタイム可視化ツール VisLink の紹介, 京都大学大型計算機センター広報, Vol. 34, No. 4, pp. 209–222 (2001).

[13] 上沢, 酒井, 小山田: リアルタイム可視化ツール VisLink によるシミュレーションステアリング, 京都大学大型計算機センター広報, Vol. 34, No. 6, pp. 327–338 (2001).

[14] 山内 聡他: アクティブボリュームレンダリングに基づくシミュレーション

ステアリング, CPSY2001-35(SWoPP 沖縄2001), pp. 1–8 (2001).

[15] Hanspeter Pfister, e. a.: The VolumePro Real-Time Ray-Casting System, *ACM SIGGRAPH 99*, pp. 251–260 (1999).

[16] 原瀬史靖: 並列ボリュームレンダリング処理の高速化, 特別研究報告書 (2002).