ボリュームレンダリング専用並列計算機

—— *ReVolver*/C40 ——

對馬 雄次

平成7年2月15日

内容梗概

ボリュームレンダリングは3次元空間を直接2次元画像に投影する手法であり、従来から、医療用の画像生成手法として重要な地位を占めてきた。さらに近年では、スーパコンピュータなど高速な計算機を利用した構造解析や流体力学など、科学技術計算の結果の解析手法としても、ボリュームレンダリングは注目を集めている。しかし、ボリュームレンダリングには膨大な計算資源を必要とするため、高速な画像生成は困難であった。

そこで、本研究では、ボリュームレンダリングを高速に行う専用並列計算機*ReVolver*/C40のアーキテクチャを提案する。*ReVolver*/C40では、従来の医療用の画像だけでなく、科学技術計算の結果の可視化をサポートする。このため、半透明ボリュームの表示、遠近法による投影、高速描画を目標としている。また、従来のサーフェスデータでは表現が困難であった霧、雲などのような表面の境界が曖昧な対象の可視化を行うため、ボリュームデータを用いたレイトレーシング法をサポートする。

これらの目標を達成するために、*ReVolver*/C40では、視線生成ステージ、ピクセル値計算ステージ、シェーディングステージからなる3ステージのマクロパイプライン構成を採用している。また、各ステージについても並列処理やパイプライン処理により画像生成の高速化を図っている。また、各ステージの主要な構成要素として汎用の高性能DSPを使用しており、柔軟性の非常に高い構成となっている。

ReVolver/C40の最大の特徴は、ピクセル値計算ステージでのボクセル・パイプライン処理である。ボクセル・パイプライン処理は、ボリュームデータを保持するメモリバンクでのボクセルの読み出しとピクセル値計算の処理を1つのステージとし、視線毎にパイプライン処理を行う方式である。このボクセル・パイプライン処理を効率良く稼働させるために、単純化されたサンプリング方法とこれに合わせたボリュームメモリ構造を採用している。これにより、任意の視線上の全ボクセルのバンクコンフリクトフリーな読み出しが可能となるので、ボクセル・パイプライン処理によるパイプラインが停滞しないことを保証できる。

また、本研究では、現在*ReVolver*/C40のプロトタイプを開発中である。このプロトタイプでは、視線生成ステージ、ピクセル値計算ステージ、シェーディングステージにそれぞれ8PE、128PE、16PEを割り当てて、処理を行う。また、プロトタイプでは、ピクセル値計算ステージのPEの構成を改良し、1PEで視線上の複数のボクセルの処理を可能としている。これにより、ボリューム空間とピクセル値計算ステージのPE数の関係が緩和され、より柔軟性を増した構成となる。

このプロトタイプの性能は、ボリューム空間が128^3voxelの時で、レンダリングの対象を不透明ボリュームとすると約68.1*ms/frame*、半透明ボリュームの場合でも、94.4*ms/frame*である。この性能は、従来最速であった専用並列計算機CUBEに匹敵するものである。さらに、視点移動に伴う再描画や任意方向からの画像生成時の性能では、CUBEの8.4倍高速である。

A Parallel Computer for Volume Rendering
— *ReVolver*/C40 —

Yuji Tsushima

Abstract

Volume rendering is a direct visualization method of three dimensional space. It is very important as a medical image generation method. Nowadays, it also acts a very powerful analysis tool for scientific computation such as structural analysis and fluid dynamics on high-performance supercomputers. Since volume rendering requiers an enormous amount of computation power and resources, it is hard to provide high-speed image generation.

This paper presents the architecture of high-speed volume rendering machine, *ReVolver*/C40. This machine visualizes not only medical data but also scientific computation results. Therefore the goals of this machine are perspective projection, translucent volume visualization, and high-speed image generation, which are useful for analysis of scientific computation results. Furthermore *ReVolver*/C40 supports ray-traced rendering of volume data. This makes it possible to visualize obscure boundary objects, such as cloud and fog, which are hardly represented in a traditional surface model.

In order to achieve these goals, *ReVolver*/C40 adopts macro-pipeline with three stages, ray casting stage, pixel calculation stage and shading stage. Parallel processing and pipelining are used on each stage for high-speed image generation. All stages mainly consist of high-performance general purpose DSPs in order todeal with various applications. The most important technique for high-speed image generation is *Voxel-Pipelining* in pixel calculation stage. A ray is pushed into the pipeline at every time step. Each stage obtains the value of a voxel on the ray in tis own volume memory bank, and calculates the contribution of the voxel to the final pixel color. *ReVolver*/C40 adopts a simplified voxel sampling method and a volume memory structure fitted for the pipeline. This makes the pipeline work efficiently without interlock, because the sampling method and the memory structure assure conflict free access to the all voxels on any ray.

We are developping a prototype machine of *ReVolver*/C40. In this prototype, ray casting stage, pixel calculation stage and shading stage have 8, 128 and 16 processing elements(PEs), respectively. The structure of PE in pixel calculation stage is improved to deal with two or more voxels on a ray. Therefore, the size of volume space is not fixed by the number of PEs in pixel calculation stage. So prototype machine has more flexible structure.

The performance of the prototype is 68.1*ms/frame* with translucent volume of 128^3 voxels and 94.4*ms/frame* with opaque volume of the same size.

1　はじめに

従来から、内部状態の可視化が可能なボリュームレンダリングは、医療用の画像生成手法として重要な地位を占めてきた。さらに近年では、スーパコンピュータなど高速な計算機を利用した構造解析や流体力学などの科学技術計算の結果が膨大に蓄積されつつあり、これらの計算結果の解析/理解の手段として、ボリュームレンダリングは注目を集めている。

しかし、ボリュームレンダリングには膨大な計算資源を必要とするため、高速な画像生成が困難であった。このため、従来からボリュームレンダリング専用のハードウェアや汎用の並列計算機上のアルゴリズムの開発により高速化が図られてきた。しかし、これらの多くは医療画像の生成を主な目的としているため、比較的小さく不透明な対象を可視化すれば十分であった。このため、平行投影法によって不透明なボリュームをレンダリングを行うためのものが主流であった。

これに対して、本論文で提案するボリュームレンダリング専用並列計算機 *ReVolver*/C40 は、医療画像と共に科学技術計算の結果の可視化を目的としたものである。科学技術計算では、地震波の解析や気象観測の結果の解析など大きな空間を扱うものがしばしばあり、遠近法による投影が必要となる。さらにボリュームデータ全体を把握するため、ボリュームを半透明に扱うことも重要である。このため、*ReVolver*/C40 では、半透明ボリュームの表示や、遠近法に基づく高速な画像生成を目標としている。この目標達成のため、ボクセルのサンプリングを単純化し、これに合ったメモリ構成を採ることにより、任意方向の視線上のボクセルの同時読み出しを可能としている。

さらに、この機構を生かしてレイトレーシング法を効果的にサポートしており、従来のサーフェスデータでは表現が困難であった、雲や霧などの表面の境界が曖昧な対象の可視化が可能となっている。

本論文では、2節で *ReVolver*/C40 の目標と目標達成のために備えている特徴について述べた後、3節で *ReVolver*/C40 の全体構成と各ステージの処理、構成について述べる。4節では、現在開発している *ReVolver*/C40 のプロトタイプについて述べ、最後に 5節でまとめる。

2　*ReVolver*/C40 の概要

本節では、*ReVolver*/C40 の目標と、その目標達成のために *ReVolver*/C40 が備えている特徴を述べる。

2.1　*ReVolver*/C40 の目標

ReVolver/C40 では、以下の 4 項目を目標としている。

目標 1. 高速描画

目標 2. 半透明ボリュームの表示

目標 3. 遠近法による画像生成

目標 4. レイトレーシング法による画像生成

以下では、これらの目標を設定した理由とこれらの目標に対する従来の対処法、問題点について述べる。

目標 1. 高速描画

ボリューム空間の断面や視点の移動に対して、対話的な速度により再描画を行うことは、可視化対象を理解/解析する上で非常に重要である。

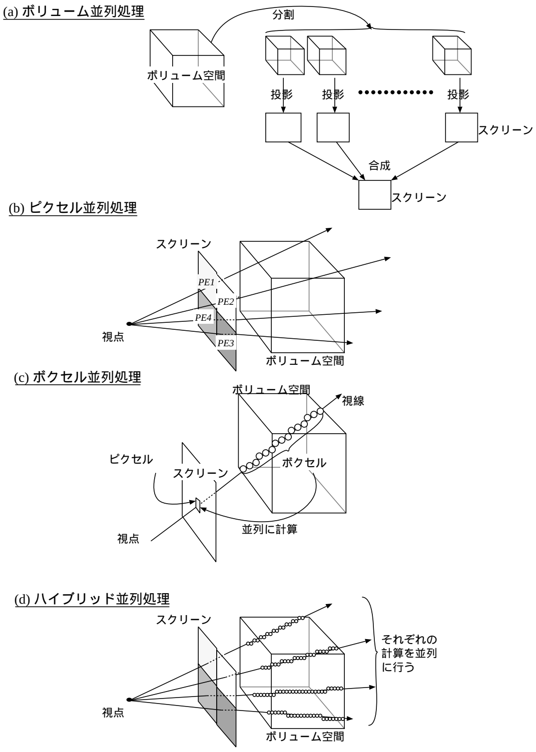


図 1: ボリュームレンダリングにおける並列処理の方式

従来から、高速描画を行うために、並列処理が行われてきた。従来行われてきた並列処理を、画像生成の手法により分類すると以下のようになる。また、図 1に、これらの概要を示す。

1. 前方投影法･･･ボリュームを構成する各ボクセルがスクリーン上の各ピクセルにどれだけ影響を及ぼすかを計算することで、各ピクセルの輝度 (以下、ピクセル値と呼ぶ) を求めるアルゴリズム。

(a) ボリューム並列処理:ボリュームをいくつかのサブボリュームに分割し、各サブボリュームに Processing Element(以下、PE と呼ぶ) を割り当てる方式。各 PE では、担当ボリューム内の各ボクセルのスクリーンへの影響を計算し、計算結果をマージすることにより最終的な画像を得る。GODPA[4],SCOPE[5],3DP^4[6] などのマシンで実現されている方式。

2. 後方投影法･･･ 視点からスクリーン上の各ピクセルを通る視線ベクトル (以下、視線と呼ぶ) を生成し、視線と交差するボクセルからピクセル値を求めるアルゴリズム。

(b) ピクセル並列処理:複数の視線が同時に処理される方式である。具体的には、スクリーンをいくつかのサブスクリーンに分割し、各サブスクリーンに PE を割り当て、ピクセル値計算を並列化する。PARCUM II[7] で実現されている方式。

(c) ボクセル並列処理:視線と交差するボクセルを全て読み出し、ピクセル値計算を多数の PE により並列に行う方式。

CUBE[8] などで採用されている。

(d) ハイブリッド並列処理:ピクセル並列処理とボクセル並列処理を融合した方式。

ここで、ピクセル値の計算の高速化のためには、同時に多数のボクセルを読み出しピクセル値計算を並列に行うことが必須がある。このため、ボリュームデータを分割し多数のメモリバンク構成としなければならない。また、後方投影法では、視線間および視線上のボクセルのサンプリングの際のバンクコンフリクトを回避しなければ、ピクセル値計算の並列処理の妨げとなり、高速化が図れない。

目標 2. 半透明ボリュームの表示

ボリュームを半透明な物体として扱うことにより、ボリューム全体の状況を把握するのに役立つ。これは科学技術計算の結果の可視化による解析手法として非常に重要である [9]。しかし、従来は半透明ボリュームをサポートしているものはわずかであった。これは、従来のボリュームレンダリングでは主に、医療分野における人体データの可視化を目的としてきたため、ボリュームを半透明として扱う必要が無かったからである。

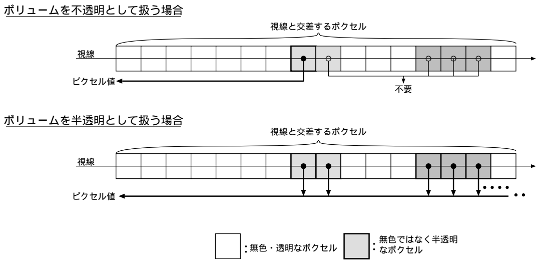


図 2: 参照するボクセルの違い

さらに、ボリュームを半透明なものとして扱うには、図2に示すように、ピクセル値計算の際に、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。したがって、読み出すボクセル数が増加し、ピクセル値計算の演算が増大するという問題が、従来の手法で半透明ボリュームのサポートされていない原因の 1 つであった

目標 3. 遠近法による画像生成

目標 4. レイトレーシング法による画像生成

遠近法によって生成した画像は、大きな空間を認識するのに役立つ [10]。また、レイトレーシングによる画像生成は、従来のサーフェスデータでは表現が困難であった、雲、霧のように境界が曖昧である対象を可視化するのに有用である事が知られている [12]。

しかし、従来の主目的である医療用の画像には、遠近法やレイトレーシング法は必要ではないため、これらの方法による画像生成を行うものはわずかであった。また、遠近法サポートする一部のシステムにおいても、視点の位置や視線の方向などに制限を設けていた。この理由は任意方向の視線を扱うと、メモリバンクへのアクセス競合の可能性が高くなり、並列処理の障害となることにある。このバンクコンフリクトの問題は、レイトレーシングにおいても同様に問題となる。

2.2 $ReVolver/\mathrm{C40}$ の特徴

$ReVolver/\mathrm{C40}$ は、2.1節で挙げた目標を達成するために、以下のような特徴を有している。

1. サンプリング方法の単純化
2. バンクコンフリクトフリーなメモリ構成
3. 並列処理による高速化

以下では、これらの特徴の各項目について述べる。

2.2.1 サンプリング方法の単純化

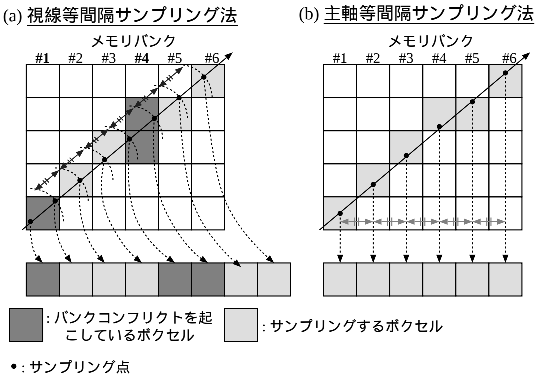


図 3: サンプリング方法の違い

従来のサンプリング方法は、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 (以下、視線等間隔サンプリング法と呼ぶ)。視線等間隔サンプリング法では図3(a) のように、ボリュームデータをメモリバンクに分割して格納すると、同一視線上の複数のボクセルが同一メモリバンクに格納されている状況が発生する。これは、ピクセル値計算の並列実行を妨げ、高速化の障害となる。

この問題を回避するために $ReVolver/\mathrm{C40}$ では、視線ベクトルの要素の絶対値のうち、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 (以下、主軸等間隔サンプリング法と呼ぶ) を採用している。この主軸等間隔サンプリング法により、図 3(b) に示すようにバンクコンフリクトが生じなくなる。

2.2.2 バンクコンフリクトフリーなメモリ構成

主軸等間隔サンプリング法では、ある座標軸に垂直な平面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平面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。

そこで、本研究で提案する 3 次元メモリ構造では、図 4 のようにボリューム空間を x, y, z軸に垂直な n 枚の平面に分割し、x, y, z- 平面群とする。ただし、n はボリューム空間の 1 辺の大きさを表している。

各平面群に対して、平面に 1,2,···,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、ボリュームデータが 3 重化されて格納されることになる。

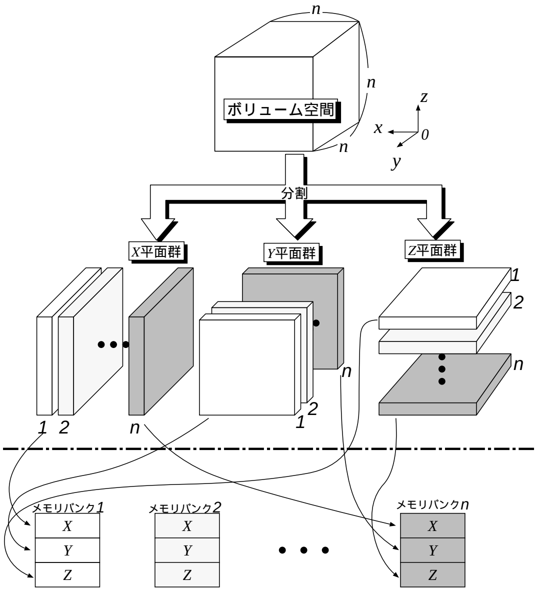


図 4: ボリュームデータの格納方法

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ているので、視線が任意方向であっても、主軸等間隔サンプリングを行うことによりバンクコンフリクトが生じないことを保証できる。これにより目標3や目標4の遠近法やレイトレーシング法のように視線の方向がピクセル毎に異なる処理に対応できる。

2.2.3 並列処理による高速化

ReVolver/C40では半透明ボリュームの表示を行うことを目標としているため、視線上のボクセルを視点に近い順に処理する必要があるが、前方投影法ではこれを保障するのが困難となる。そこで、*ReVolver*/C40は比較的容易にこれを実現できる後方投影法を採用する。

また、並列化の手法としてピクセル並列処理、ハイブリッド並列処理では、複数視線の処理を同時に行うため、ボリュームメモリへのアクセスが競合する可能性が高くなる。単純なボクセル並列処理では、主軸等間隔サンプリング法と3次元メモリにより、視線と交差する全ボクセルの同時読み出しが可能である。しかし、ボクセルを同時に読み出すためには、視線情報をブロードキャストし、並列にアドレス計算を行う必要がある。さらに、読み出したデータを並列処理するために木構造のピクセル値計算機構が必要になる[1][3]。そこで、本研究では、ピクセル並列処理とボクセル並列処理を融合したボクセル・パイプライン処理を考案した。これは図5に示すように、視線を順に投入し、各視線は、1つのメモリバンクからのボクセルの読み出しとそのボクセル値を用いたピクセル値計算を1つのステージとしてパイプライン処理を行う方式である。この方式は、ある時刻でのパイプラインの状態を見れば、ピクセル並列処理を行っているが、主軸等間隔サンプリング法と3次元メモリ構造を用いることによりバンク競合が起こり得ないため、パイプラインが停滞しないことを保証できる。

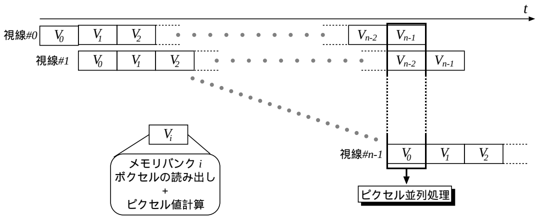


図 5: ボクセル・パイプライン処理

3 *ReVolver*/C40の処理と構成

3.1 全体構成

本節では、2.2節で示したような特徴を備えた*ReVolver*/C40の構成要素とその機能、特徴について述べる。また、これらの要素間、要素内のPEの相互結合については図6に示す通りである。

- **SCU**(*Ṡystem Ċontrol U̇nit*): *ReVolver*/C40とホストマシンの中間に位置しており、ホストマシンからのボリュームデータ/制御コマンド/各種パラメータを各部に伝達する。
- **RCS**(*Ṙay Ċasting Ṡtage*): RCSは図6に示すように、多数のPEが並列に配置されている。RCSでは、SCUから送られてくる視点、スクリーンの情報から、スクリーンを分割してPEに割り付け、視点から各ピクセルを通る視線ベクトル(以下、視線と呼ぶ)の生成を並列に行う。
- **PCS**(*Ṗixel Ċalculation Ṡtage*): PCSは図6に示すように、PEが1次元に接続された構造であり、各PEは、ボリュームデータ、ボクセル値を色、透明度に変換するための属性テーブルを保持している。各PEの処理は、RCSから送られてくる視線データによりボクセルをサンプリングし、ピクセル値計算を行うことである。このピクセル値計算をPE間でパイプライン実行する。
- **SS**(*Ṡhading Ṡtage*): SSは図6に示すように、PEは画像処理に適したメッシュ状に配置されており、シェーディング処理などのフィルタ処理をスクリーン分割により並列実行する。

  SSの各PEが行う処理は、PCSが生成した画像に対してdepth gradient shadingを施し、フレームバッファへ送出することである。また、SSでは画像のクリッピング処理や拡大・縮小の処理も行う機能を持つ。

ReVolver/C40では、各ステージの主要部品として、汎用の高性能DSP (TMS 320C40GFL-50)を使用することにより、多様な処理に対して柔軟に対応可能な構成となっている。

ここで、このDSPの特徴[19]は、図7に示すように、転送速度$100MB/sec$のGlobal BusとLocal Busを持ち、さらに1本の$20MB/sec$の転送速度を持つcom portを6本持つ点である。このうち、Local Bus、Global Busはボリュームメモリ、DSP間の通信のためのバッファが接続されており、com portはステージ間通信やDSP間通信のために使用されている。また、このDSPは、乗算と加減算、加減算とロードストアを1命令として実行する機構を持ち、DSPの実行命令数の削減に役立っている。

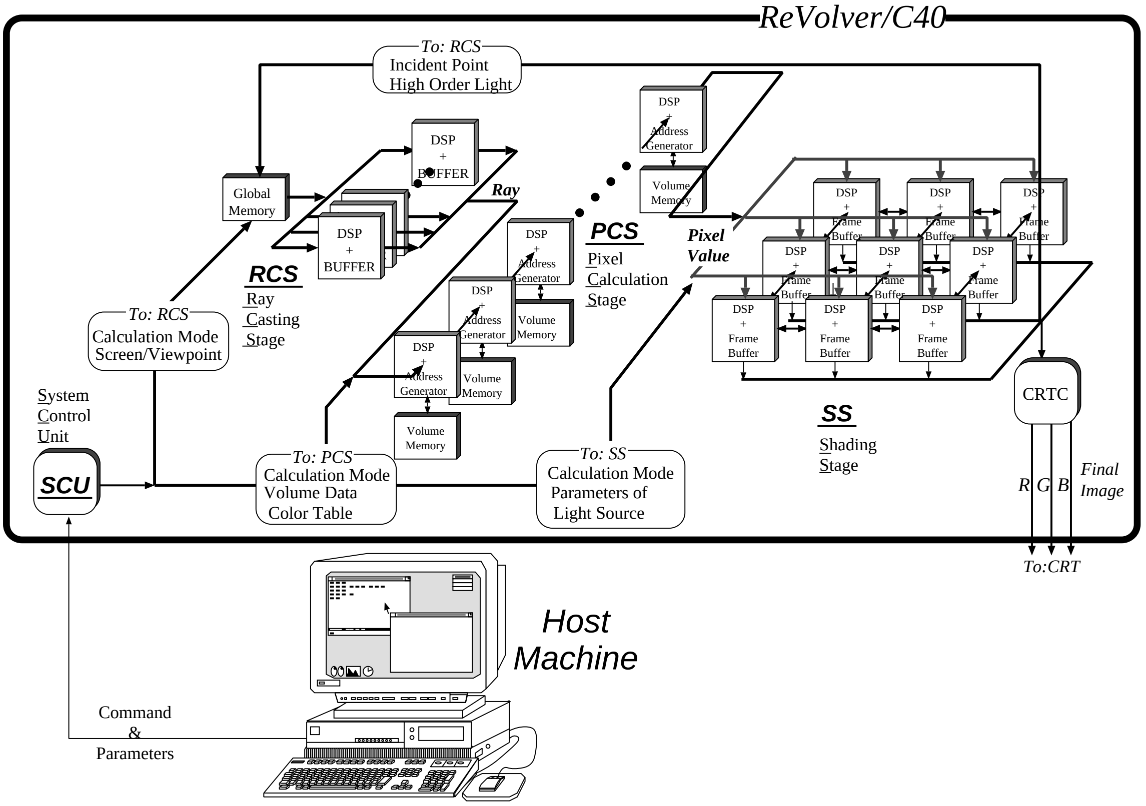
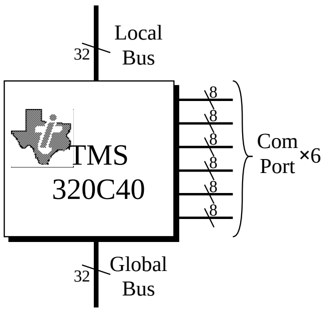


図 6: $ReVolver$/C40 の全体構成



- Local/Global Bus: 最大転送速度100MB/sec
- Com Port: 1本で最大転送速度20MB/sec

図 7: TMS320C40 の特徴

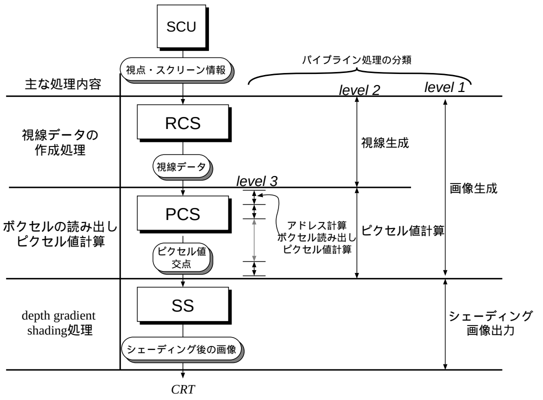


図 8: 通常モードの動作

なお、以下本論文では、$ReVolver$/C40 が扱うボリューム空間、スクリーンの大きさは、それぞれ $512^3 voxel$、$512^2 pixel$であるとする。

3.2 処理の概要

3.2.1 通常モードの処理

$ReVolver$/C40 の通常モードでの各ステージ間で流れるデータと主な処理を図 8に示す。また、$ReVolver$/C40 で行われているパイプライン処理を、流れるデータにより分類すると次のようになる。

level 1. RCS と PCS による画像生成と、SS によるシェーディング、画像出力からなるフレーム単位のパイプライン処理。

level 2. RCS の視線生成と PCS によるピクセル値計算の視線単位のパイプライン処理。

level 3. PCS の 1PE が 1 ステージであり、ピクセル値を単位としたパイプライン処理。

以下では、これらのパイプライン処理における各ステージの処理と並列処理の手法について述べる。

- **RCS の処理:** RCS の処理は、SCU から受け取った視点、スクリーンの情報を基にして視線生成を行うことである。RCS で行われている並列処理の方法は、スクリーンを横方向に PE 台数で分割し、静的に各 PE に割り付け、各 PE は担当領域の視線の生成を並列に行う。

ここで、横方向に分割している理由は、視線生成処理において、スキャンラインの左端のピクセルについて視線を

生成すれば、残りのピクセルの視線にはスクリーンの水平方向の単位ベクトルを加えればよいので、DSP の命令数の削減に役立つからである。

生成した視線は PCS へ 1 つずつ送信されて、ボクセルのサンプリングのためのアドレス生成に使用される。

・**PCS の処理:** *ReVolver*/C40 では、ボリュームを半透明なものとして扱っているため、PCS で行われるピクセル値計算では、視線上のボクセルの値を視点に近い順に $v_0, v_1, \cdots, v_{511}$ とすると、以下の式 (1) を評価した値をピクセル値としている [9][14]。ただし、この式では、$c(v_i), \alpha(v_i)$ はボクセル値 v_i を色、透明度に変換した値であることを示している。

$$\text{ピクセル値} = \sum_{i=0}^{511}(1-\alpha(v_i))c(v_i)\prod_{j=0}^{i-1}\alpha(v_j) \qquad (1)$$

この処理と同時に、PCS では SS で行われるシェーディング処理の前処理として、視線上で視点に最も近く無色・透明でないボクセルを読み出した PE の番号を求め、SS へ送信する。

この処理を PCS では、式 (1) の評価と交差判定を、1 次元に接続された PE によりパイプライン処理している。

・**SS の処理:** SS の処理は、PCS で計算されるピクセル値と交点を受け取りフレームを構成し、このフレームに対して SCU からの光源情報を用いて、 depth gradient shading を施し、CRT へ出力することである。ただし、PCS からの送られてくるデータは、SS で処理される時のデータと型が異なるため、受信時に変換する。また、SS は生成した画像に対してクリッピングや拡大・縮小の処理も行う。

この処理を SS では、スクリーンを分割し PE に割り付けることで並列処理を行っている。

3.2.2 通常モードのオプション

本節では、通常モードで *ReVolver*/C40 が用意しているピクセル値計算のオプションについて述べる。

1. 不透明ボリュームの表示

ReVolver/C40 では、従来のボリュームレンダリング処理の使用目的である医療画像の生成にも対処するため、ボリュームを不透明なものとして扱うオプションを用意している。この際の処理は、視線上のボクセルで視点に最も近く、無色・透明でないボクセルの値のみからピクセル値を求めることになる。

また、*ReVolver*/C40 は任意方向の視線上のボクセルの同時読み出しを、バンクコンフリクトフリーで行えるため、投影方法については遠近法、平行投影法のどちらでもよい。

2. 連続モデルに基づくボクセルによるレンダリング

通常 *ReVolver*/C40 で扱っているボクセルでは、ボリューム空間を構成する単位立方体で、中心に値を持ち単位立方体内の全ての点は中心と同じ値を持つとする離散モデルであるが、連続モデルに基づくボクセルを用いてレンダリングすることで高精細な画像の生成が可能となる。連続モデルに基づくボクセルとは、単位立方体の頂点にボクセル値が存在し、単位立方体の内部の点の値は、その点を囲む 8 頂点の値から tri-linear 補間 (図 9参照) により求められるものである [9]。

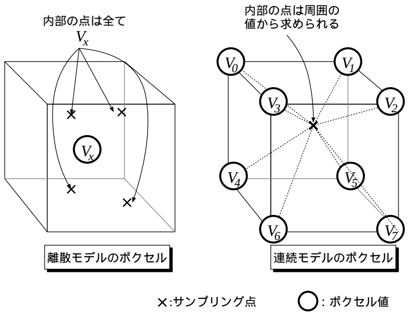


図 9: ボクセルのモデル

$$\left.\begin{array}{rcl} C_i & = & (1-x_i-y_i-x_i\cdot y_i)\cdot c_{(i,0)} \\ & & +x_i\cdot(1-y_i)\cdot c_{(i,1)} \\ & & +y_i\cdot(1-x_i)\cdot c_{(i,2)} \\ & & +x_i\cdot y_i\cdot c_{(i,3)} \\ \alpha_i & = & (1-x_i-y_i-x_i\cdot y_i)\cdot \alpha_{(i,0)} \\ & & +x_i\cdot(1-y_i)\cdot \alpha_{(i,1)} \\ & & +y_i\cdot(1-x_i)\cdot \alpha_{(i,2)} \\ & & +x_i\cdot y_i\cdot \alpha_{(i,3)} \end{array}\right\} \qquad (2)$$

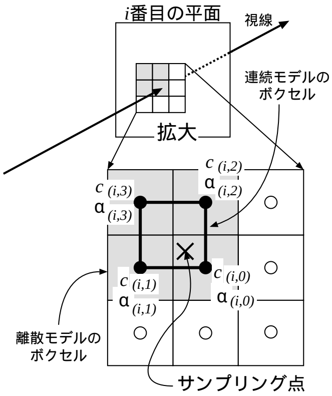


図 10: *ReVolver*/C40 で行う tri-linear 補間

ここで、主軸が z 軸であり、サンプリング点の座標の小数部が (x_i, y_i) である時、*ReVolver*/C40 で行われる補間を図 10中の式 (2) に示す。ただし、式 (2) では図 10で示される記号を使用している。*ReVolver*/C40 では、主軸等間隔サンプリング法を採用しているため、サンプリング点は必ず単位立方の側面上に位置することになる。このため、サンプリング点を含むボクセルの側面上の 4 頂点のみで、tri-linear 補間が行える。したがって、従来の視線等間隔サンプリング法による補間と比べて演算量・メモリアクセスが半分となり、高速実行が可能となる。

3. 主軸等間隔サンプリング法のサンプリング間隔の補正

ReVolver/C40 で採用している主軸等間隔サンプリング法は、主軸が座標軸と平行でない視線では、従来の視線等間隔サンプリング法に比べて、サンプリング点の間隔が大きくなる傾向があり、最大で $\sqrt{3}$倍となる。そこで、サンプリング間隔の違いを、色・透明度を調整することにより補正するオプションを用意する。

$ReVolver$/C40 が採用している補正方法は、補正前の色・透明度をC, α、補正後の色・透明度をC', α'、主軸等間隔サンプリング法によるサンプリング間隔をΔDとすると、以下のように表すことができる [3]。ただし、$0 \leq \alpha \leq 1, 1 \leq \Delta D \leq \sqrt{3}$である。

$$C' = C \cdot \alpha^{\Delta D-1} \qquad \alpha' = \alpha^{\Delta D}$$

しかし、これらの式を直接求めるのは、PE への負担が大きいため、直接評価せず式 (3) の 2 次の近似式により$\alpha^{\Delta D}$を求めることにしている。

$$\alpha^{\Delta D} = 1 + \Delta D \cdot (\alpha - 1) + \frac{\Delta D \cdot (\Delta D - 1)}{2} \cdot (\alpha - 1)^2 \quad (3)$$

3.2.3 レイトレーシングモード (RT モード) の処理

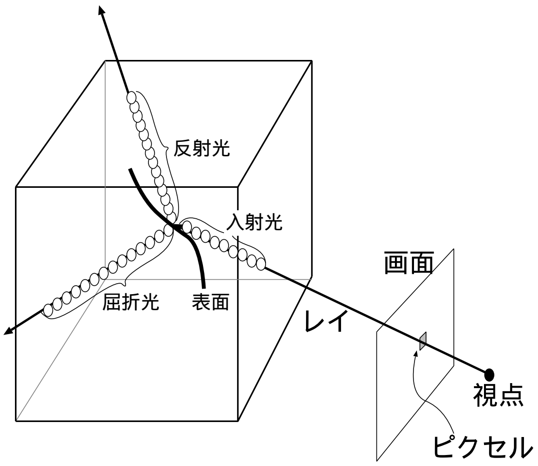


図 11: レイトレーシングの様子

ボリュームデータを使ったレイトレーシングは、図 11に示すように、視線と表面が交差すると反射・屈折光 (以下、高次光とよぶ) を生成し、再帰的に処理するものである。そこで、$ReVolver$/C40 のレイトレーシングモード (以下、**RT** モードと呼ぶ) では、図 12 のように SS が高次光を生成し、RCS へ高次光を送り返して視線として処理することで、レイトレーシングを行っている。

以下では、各ステージの処理の通常モードと異なる点について述べる。

·**RCS の処理:**通常モードの処理に加えて、SS から送られてくる高次光も視線として PCS へ送信する。

·**PCS の処理:**RT モードにおける PCS の処理で、通常モードと異なるのは、ピクセル値計算で視線が表面と交差する場合、交点の手前までの値をピクセル値としなければならない点である。さらに、通常モードと異なり、以下で述べる 2 つの処理も行う。

1. 交差判定··· 従来のサーフェスデータとは異なり、ボリュームデータはオブジェクトの境界面の決定が困難なため、視線と表面の交差判定も困難となる。また、通常モードと違い、視線と交差する最初の無色・透明でないボクセルを交点とする交差判定では、高次光が連続して反射屈折を繰り返すため使用できない。

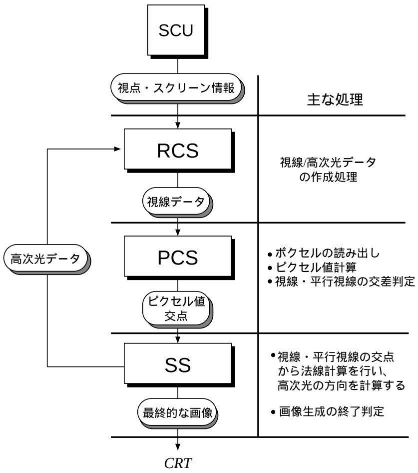


図 12: RT モードの動作

そこで、RT モードでは視線上で隣接するボクセル値の差を一定値δと比較し、この差が大きい場合、交点の候補として選択する。さらに、これら候補の中から視点から最も近いものを選ぶことにより交差判定を行う。

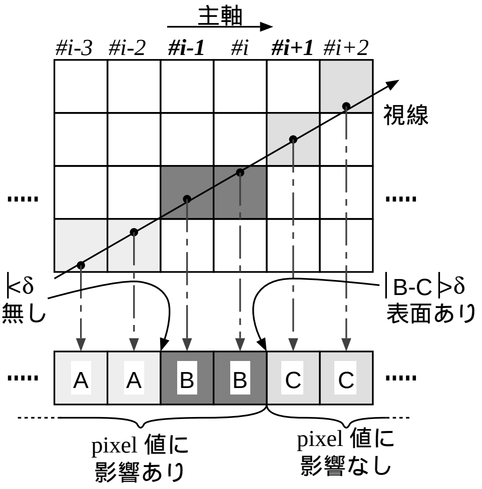


図 13: 視線と表面の交差判定

図 13の例では、$\#i-1$ と$\#i+1$ で隣接ボクセル値が異なるが、$\#i-1$ では一定値δ未満で、$\#i+1$ ではδより大きい。この時、$\#i+1$ が表面として選択される。また、ピクセル値の計算は、$\#i$ で打ち切られ、以降のボクセルはピクセル値へ影響を及ぼさないようにしなければならない。このピクセル値計算の打ち切りについては、PCS のk番目の PE が交点であると判定したとすると、主軸の方向が正の場合は $k+1$ へ渡す透明度を 0 とし、負の場合 $k+1$ 以降へ渡すピクセル値を 0 とすればよい。

2. 法線計算の前処理:··· 交点での法線ベクトルは高次光の方向計算に必要不可欠である。従来の法線計算の方法は、各ボクセルに存在濃度を設定し、交点周辺の存在濃度から交点での法線を計算していた [15]。

従来の方法で問題となるのは、ボリュームデータが分割されて PE 毎に配置されている場合、交点のボクセルを保持している PE が大量のメモリアクセスと法線を求めるための計算を行うことである。したがって、このような PE

によりシステム全体の速度が律速される危険性がある。

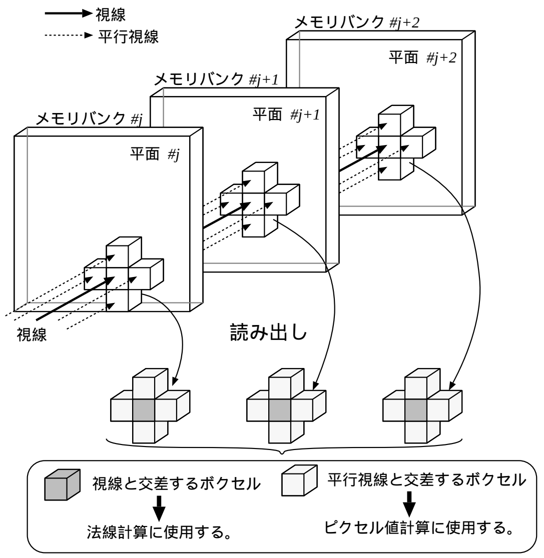


図 14: RT モードのメモリアクセスパターン

そこで本研究で提案する方法は、この問題を回避するために、視点が主軸以外の軸方向に移動した点から視線と平行な視線 (以下、平行視線と呼ぶ) が出ているものと考え、各平面では平行視線上のボクセルもサンプリングする (図14参照)。ただし、視点の移動距離に関しては、可視化対象の特徴を考慮する必要があるので、SCU からのパラメータにより指定できるようにしなければならない。

この平行視線に対しては、視線と同様の方法による交差判定のみを行うようにする。したがって、レイトレ時は視線、平行視線から 5 つの交点が求められ、これらを A、W、E、N、S とする (図 15参照)。この時の点 A での法線ベクトルは、以下の式 (4) により計算できる。ただし、式 (4) では$\vec{A}\times\vec{B}$は$\vec{A}$、$\vec{B}$の外積を表している。

$$\overrightarrow{\text{法線}} = \overrightarrow{AN}\times\overrightarrow{AW}+\overrightarrow{AW}\times\overrightarrow{AS}+\overrightarrow{AS}\times\overrightarrow{AE}+\overrightarrow{AE}\times\overrightarrow{AN} \quad (4)$$

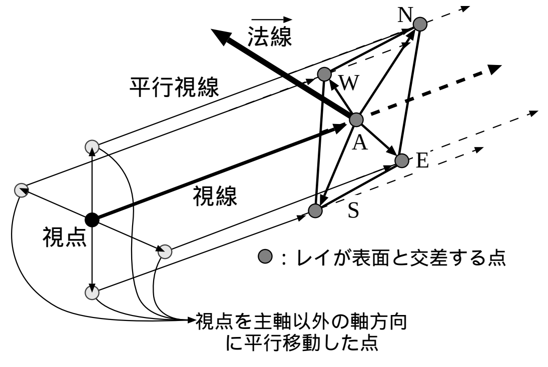


図 15: 法線の計算方法

従来の方法で、交点を中心として 1 辺が $7voxel$のサブキューブから法線計算を行うとすると、法線計算のために $7^3 - 1 = 342voxel$を余分に読み出すことになる。一方、本手法では平行視線の分を余計に読み出すので、最大 $512 \times 4 = 2048voxel$になり、全体で読み出すボクセル数は増加している。しかし、交点を含む平面から読み出すボクセルの数は、従来の方法で $7^2 = 49voxel$なのに対して、本手法では $5voxel$である。これは、法線計算のためのボクセルを視線に沿って均等にアクセスしているためであり、メモリアクセスと法線計算処理が均等に分散されていることを表している。

- **SS の処理:**通常モードではシェーディング処理を行っていたが、シェーディング処理はレイトレーシング法に含まれているので、RT モードではシェーディングを行う必要がない。そこで、RT モードでは、PCS から視線と平行視線の交点を受け取り、これらのボリューム内座標から、視線の交点での法線を式 (4) により計算し、これを用いて高次光の方向の計算を行う。生成した高次光のデータは RCS へ送信される。

さらに、SS の処理として、画像生成の終了判定がある。画像生成の終了条件は、全ピクセルが、一定回数の高次光を生成、あるいは、高次光を生成しない状況になることであり、SS の PE は担当領域のピクセルに関して、終了条件が満たされているかどうかを調べる処理を行う。

3.3 SCU(System Control Unit)

現在広く使われている PCI または EISA を装備したワークステーションをホストマシンとして使用することを想定しており、このバスを通してホストマシンとの間でデータ、コマンド等の授受を行う。

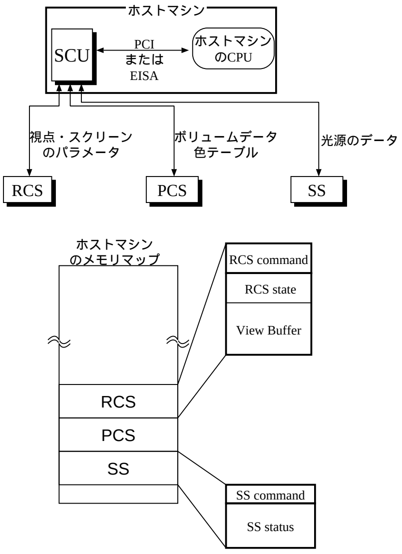


図 16: SCU の構成

また、SCU と他の構成部分との接続は図 16に示すように、各ステージとの間に個々にリンクを設けており、SCU が各ステージと送受信するデータは表 1で表されるものである。

以下、SCU と各ステージ間の通信方法について述べる。

SCU と RCS 間の通信

RCS の全ての PE で共有されている View Buffer(3.4.3 参照) をホストマシンのメモリにマップし、ホストマシンから直接読み出し/書き込みが行うことで実現されている。この View Buffer の一部を用いることで、RCS のステータスや、DSP のプログラムの授受を行う。

表 1: SCU と他の構成部分間のデータ

SCU と RCS 間のデータ		
データ名	データ転送の方向	データサイズ
視点	SCU → RCS	32bit float × 3
スクリーン	SCU → RCS	16bit float × 9
RCS コマンド	SCU → RCS	——
DSP のプログラム	SCU → RCS	——
RCS ステータス	RCS → SCU	——

SCU と PCS 間のデータ		
データ名	データ転送の方向	データサイズ
DSP のプログラム	SCU → PCS	——
色テーブル	SCU → PCS	16bit float × 256 × 3(4)†
ボリュームデータ	SCU → PCS	512^3 byte
PCS コマンド	SCU → PCS	——
PCS ステータス	PCS → SCU	——

SCU と SS 間のデータ		
データ名	データ転送の方向	データサイズ
光源位置	SCU → SS	32bit float × 3
光源色	SCU → SS	16bit float × 3
SS コマンド	SCU → SS	——
SS プログラム	SCU → SS	——
SS ステータス	SS → SCU	——

——: 可変長のデータであることを表す。
†: () 内はレイトレ時の数字。

SCU と PCS、SS の通信

各ステージに代表の PE を設けて、この PE の DSP が持つ com port を使って SCU と各ステージの通信を実現している。したがって、SCU は、代表の PE に対してのみデータを送信し、残りの PE に対しては、各ステージの持つ結合網によりデータ送信することで全 PE への通信を実現する。

3.4 視線生成ステージ (RCS)

本節では、RCS で行う処理の詳細を通常モードと RT モードに分けて述べた後、RCS の構成について述べる。

3.4.1 通常モードの RCS の処理

通常モードの RCS の処理は、SCU から与えられる視点、スクリーンのパラメータからスクリーン上のピクセルを通る視線ベクトルを作成し、ボクセルのサンプリング時に必要な情報を付加して PCS へ送信することである。

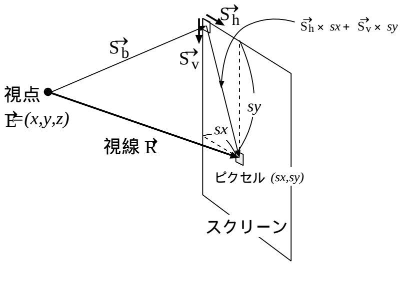


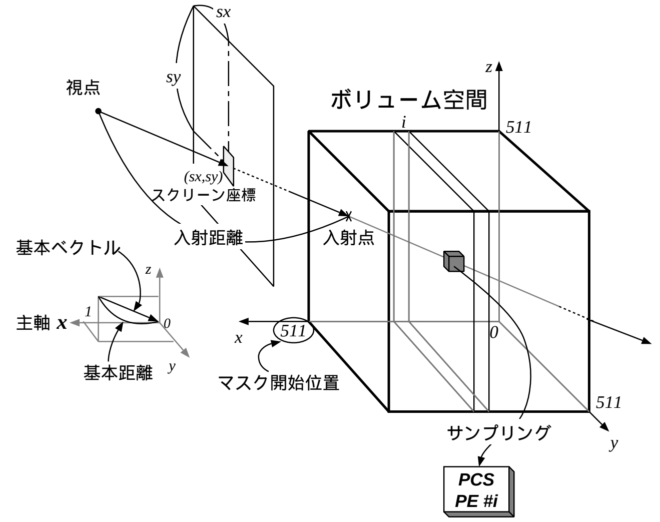
図 17: SCU から RCS へのパラメータ

SCU から受け取る、視点、スクリーンのパラメータは以下の通りであり、各パラメータ間の位置関係を図 17に示す。

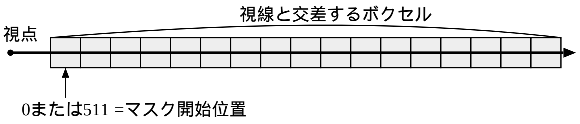
- 視点 ··· 視点の座標を表す。このデータはボリューム空間の 1 頂点を原点に持つワールド座標系で与えられる。
- スクリーン ··· 以下の 3 つのベクトルにより表される。

$\vec{S_v}$: スクリーンの水平方向の単位ベクトル。
$\vec{S_h}$: スクリーンの垂直方向の単位ベクトル。
$\vec{S_b}$: 視点からスクリーンの右上頂点へのベクトル。

これらのデータの相互関係を図 18で示す。以下では、これらのデータを使用して RCS が生成する視線データ (下線により示す) について述べる。



(1) 視点がボリュームの外部にある場合



(2) 視点がボリュームの内部にある場合

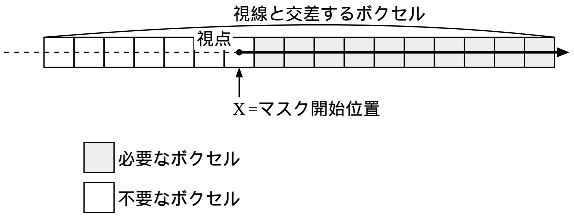


図 18: RCS で作成する視線パラメータの相互関係

PCS で使用されるデータ:PCS の i 番目の PE がサンプリングするボクセルのアドレスは以下の式 (5) のように表される。このため、RCS から PCS へ渡されるデータに式 (5) の入射点、基本ベクトルが必要となる。入射点は図 19に示すように、主軸方向に視線を延長した時に最初にボリューム空間の幅 (0 ~ 511) に入射する際の座標であり、基本ベクトルは視線ベクトルの主軸成分を 1 とした時の視線ベクトルである。

$$\text{入射点} + \text{基本ベクトル} \times i \tag{5}$$

さらに、PCS で保持しているボリュームデータは 3 重化されているため、PCS は RCS から受け取る主軸名により、ボクセルを読み出すメモリ領域の選択を行わなければならない。また、PCS では、視線ベクトルの主軸成分の符号によりピクセル値計算の方法を変更しなければならないので、主軸方向を RCS から受け取る。

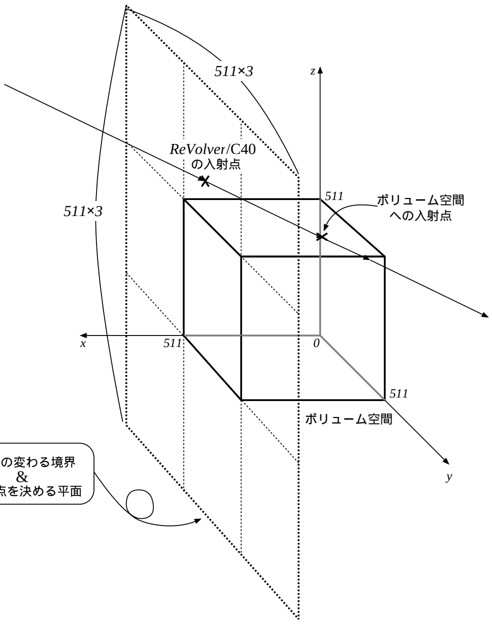


図 19: 入射点の位置

また、視点がボリュームの内部にある場合やボリュームの切断面の表示を行うために、図 18の (1)、(2) のように読み出されたボクセルが不要かどうかを判定しなければならない。そこで、RCS はマスク開始位置により、これを知らせるようにする。

SS で使用されるデータ: SS では、PCS からピクセル値を受け取る際に、フレームバッファへ格納するためのアドレスの生成を行わなければならないが、これにはスクリーン座標を RCS から SS へ伝える必要がある。また、SS で行われる depth gradient shading に必要な Z 値は以下の式 (6) で求められる。ここで、入射距離、基本距離は、それぞれ視点から入射点までの距離、基本ベクトルの長さを表しており、これらは、RCS で計算され、SS へ渡される必要がある。

$$\text{入射距離} + \text{交点} \times \text{基本距離} \tag{6}$$

しかし、RCS は、SS への直接のリンクを持たないため、これらのデータは PCS を経由して SS へ渡されることになる。

これらのデータの作成手順と各データの記号名を、SCU から受け取るパラメータの表記を用いて表 2に示す。ただし、表 2では主軸を x 軸とし、ベクトル$\vec{A}$の要素を $A.x, A.y, A.z$ として表している。

3.4.2 RT モードの RCS の処理

RT モードの RCS の処理は、3.4.1節で述べた処理に加えて、SS から送られてくる高次光のパラメータから視線データを作成し PCS へ送信することである。

シェーディングステージから送られてくる高次光のデータは以下の通りである。

- 高次光の方向 ··· 高次光の方向ベクトル。
- 交点 ··· 視線がボリューム内のオブジェクトと交差する点のボリューム内座標。
- スクリーン座標 ··· 高次光を発生させた視線または高次光のスクリーン座標。

高次光のための視線データで通常モードと異なるのは、RT モードではシェーディング処理を行わないため、シェーディング処理に必要な入射距離、基本距離の作成を行わないことである。しかし、PCS の処理では高次光と視線を区別しないため、入射距離、基本距離にはダミーのデータを送る必要がある。また、高次光の入射距離、基本距離以外の視線データは、高次光の方向と交点をそれぞれ表 2の $\vec{R}$、$\vec{E}$と置き換えて、表 2で示した通常モードの視線データ作成の手順により作成される。

なお、RCS の各 PE では高次光の作成を通常の視線生成処理より優先して作成するようにする。これは、通常の視線を優先させると、SS から送られてくる高次光のデータが急激に増大するためである。

3.4.3 RCS の構成

RCS は、以下の要素から構成されており、図 20に示すように他のステージと接続されている。

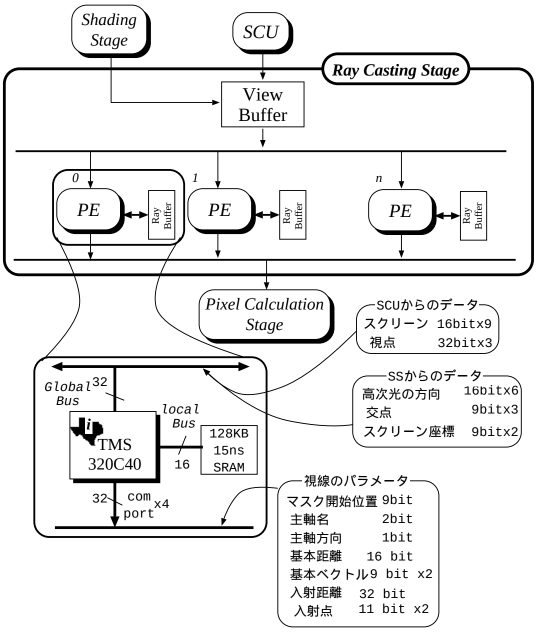


図 20: RCS での PE の接続

- **PE:** View Buffer で保持されている視点、スクリーンのパラメータ/高次光のデータを受け取り視線のデータ作成を行うための DSP。
- **Ray Buffer:** PCS への送信する視線データを保持するためのバッファ。DSP の Global Bus に接続される。
- **View Buffer:** RCS の全 PE で共有され、SCU から送られてくる視点、スクリーンの情報や SS から与えられる高次光のデータを保持するためのバッファ。DSP の Local Bus に接続される。

PCS と RCS は、DSP の com port の 4 本を使用して、共有バスを構成し PCS の 0 番 PE と接続する構成としている。RCS では、生成した視線データを全て、一旦 Ray Buffer へ格納する。1 スキャンライン分の視線データの生成が終わると、共有バスの獲得要求を出し、獲得できれば

表 2: RCS でのデータ作成手順

データ名 (記号名)	作成方法	bit 数
主軸名 $(AXIS)$	$AXIS = \begin{cases} if(\text{主軸} == x)\ 01 \\ if(\text{主軸} == y)\ 10 \\ if(\text{主軸} == z)\ 00 \end{cases}$	$2bit$
主軸方向 (DIR)	$DIR = (R.x > 0)?\ 0 : 1$	$1bit$
基本ベクトル $(BV1, BV2)$	$\vec{R} = \vec{E} + \vec{S_b} + sx \times \vec{S_h} + sy \times \vec{S_v}$ $BV1 = R.y/\lvert R.x\rvert;$ $BV2 = R.z/\lvert R.x\rvert$	$9bit \times 2$
基本距離 $(BLEN)$	$BLEN = \sqrt{BV1^2 + BV^2 + 1}$	$32bit$
入射点 $(IN1, IN2)$	if 511> $E.x$ >0 then $\begin{cases} IN1 = E.y; \\ IN2 = E.z; \end{cases}$ else $\begin{cases} IP1 = E.y - E.x \times BV1; \\ IP2 = E.z - E.x \times BV2; \end{cases}$	$12bit \times 2$
入射距離 $(ILEN)$	$ILEN = \sqrt{(DIR \times 511 - E.x)^2 + (IP1 - E.y)^2 + (IP2 - E.z)^2}$	$32bit$
スクリーン座標 $(PXL.x, PXL.y)$	$PXL.x = sx$ $PXL.y = sy$	$9bit \times 2$
マスク開始位置 $(MASK)$	if $0 < E.x < 511$ then $Mask = E.x$ else $Mask = (MD > 0)?\ 511 : 0$	$9bit$

PCS への送信を開始する。獲得できない場合は、別のスキャンラインの視線データを作成し続ける。また、共有バスが空くと共有バスコントローラが、他の送信可能な PE へ割り込みをかけ、PCS への送信を開始させる。

ここで、表 2に示した通常モードの視線データ作成手順に必要な命令数を求めたところ、1 視線あたり 67 命令であることが分かった。今回使用する DSP のサイクルタイムが $20ns$ であるから、仮に視線生成をリアルタイム $(33.3msec/frame)$ で行うとしても、約 10 個程度の DSP を配置すれば充分ということになる。

また、View Buffer はバスにより PE と結合されているが、以下の理由により View Buffer へのアクセス頻度が低く、バス競合により PE の並列動作が妨げられることはないためである。この問題については、4.2.2節で詳細に述べる。

- View Buffer は DSP の Local Bus に接続されており、Local Bus は最大 100MB/sec のバンド幅を持つ。
- 各 PE からのアクセスは 1 フレームに 1 回である。
- PE 数は高々 10 個程度と少数である。

3.5 ピクセル値計算ステージ (PCS)

PCS は 2.2.2節で示した方法により分割したボリュームデータを格納しているステージであり、視線と交わるボクセルを読み出しピクセル値の計算を行う。本節では、PCS で行う処理の詳細を通常モードと RT モードに分けて述べた後、PCS の構成について述べる。

3.5.1 通常モードの処理

RCS からの視線データを基にして、その視線上にあるボクセルをボリュームメモリから読み出し、このボクセルの値を用いてボクセルの色、透明度を求めて、ピクセル値を計算する。

ここで、PCS から SS へ送信されるデータについて以下にまとめる。これらは SS でシェーディング処理を施された後 CRT へ出力される。

- ピクセル値: $ReVolver$/C40 では、半透明ボリュームの可視化を行うために、視線上のボクセルの値を視点に近い順に $v_0, v_1, \cdots, v_{511}$とすると、3.2.1節の式 (1) により計算される。ここで、以下のような式を定義する。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。

$$\begin{aligned} C_k &= \sum_{i=0}^{k}(1-\alpha(v_i)) \cdot c(v_i) \cdot \prod_{j=0}^{i-1}\alpha(v_j) \\ A_K &= \prod_{i=0}^{k}\alpha(v_i) \\ D_k &= \sum_{i=k}^{511}(1-\alpha(v_i)) \cdot c(v_i) \cdot \prod_{j=k}^{i-1}\alpha(v_j) \end{aligned}$$

ここで、式 (1) は、C_{511}または D_0のと同一の式である。また、これらの式は、以下のような漸化式により表される。

$$\begin{cases} C_k = C_{k-1} + (1-\alpha(v_k)) \cdot c(v_k) \cdot A_{k-1} \\ A_k = \alpha(v_k) \cdot A_{k-1} \end{cases} \quad (7)$$

$$D_{k-1} = D_k \cdot \alpha(v_{k-1}) + (1-\alpha(v_{k-1})) \cdot c(v_{k-1}) \quad (8)$$

したがって、ピクセル値計算を行うには、式 (7) または式 (8) を順に計算し、C_0または D_{511}を求めればよいことになる。そこで、式 (7) または式 (8) の 1 回分の処理を、PCS の 1PE により行うようにする。

ここで、式 (7) と式 (8) ではボクセルのアクセス順序が反対であることから、主軸方向が正の時は式 (7)、負の時は式 (8) を各 PE で計算するようにすれば、いずれの場合も 0 番 PE から処理を開始できることになる。したがって、RCS のデータは PCS の 0 番 PE へ入力され、ピクセル値が 511 番 PE から出力されることになる。

- **交点:** 視線上で視点に最も近く無色・透明でないボクセルの座標の主軸成分。シェーディングステージでのピクセルの傾斜計算に使用される。
- **スクリーン座標、基本距離、入射距離:** これらは RCS で作成されるものであるが、PCS を経由して SS へ渡される。

式 (1) で使用している $c(v_i)$、$\alpha(v_i)$ は、読み出したボクセル値 v_iを使って属性テーブルを参照することにより求められる。ボクセル値は $1byte$ で表されているため、このテーブルは 256 エントリから構成される。各エントリの構成は$c(v_i) = (R(v_i), G(v_i), B(v_i)), \alpha(v_i)$ からなり、各データは $16bit\ float$ で表わされている。このテーブル参照は、以下の 2 通りのうちいずれかで行われる。

- **直接:** ボクセル値を直接インデックスとして、属性テーブルを参照する。 通常はこちらを使用する。
- **差分:** 隣接ボクセル値との差分をインデックスとして、属性テーブルを参照する。ボクセル値の変化が急な箇所を強調するのに使用される。ただし、ボクセル値の差分は$-255 \sim 255$ までの値をとり得るため、差分の値を $1bit$ 右シフトし 128 を加えたものをインデックスとする。

3.5.2 RT モードの処理

3.2.3節で述べたように、RT モードの PCS はピクセル値計算に加えて、交差判定、法線計算の前処理を行う。ただし通常モードと異なり、ピクセル値計算については、交点の手前で打ち切らなければならない。そこで、k 番 PE が交点であると判断したとすると、主軸方向が正であれば式 (7) で $A_{k-1} = 0$、負であれば式 (8) で $D_{511-k} = 0$ としてピクセル値計算を行えば、交点の手前で打ち切る処理と同じ効果が得られる (図 21参照)。

ここで、RT モード時に PCS から SS へ送信されるデータをまとめると以下のようになる。ただし、SS へ渡されるデータで種類や個数が通常モードと異なるものは、下線により表されているものである。

- 色 $\cdots$ 視線の交点を kとすると、主軸の正負に応じて、式 (7) の C_{k-1}または D_{511-k}をピクセル値とする。
- 交点 (5 つ) $\cdots$ 視線、平行視線ぞれぞれの交点。これらを用いて、SS で交点の法線計算が行われる。
- ボクセル値 (2 つ) $\cdots$ 交点の直前のボクセル値と交点のボクセル値。SS で屈折率へ変換するのに使用される。
- スクリーン座標、入射点、基本ベクトル $\cdots$ RCS で作成されるデータが PCS を通過して SS へ渡される。

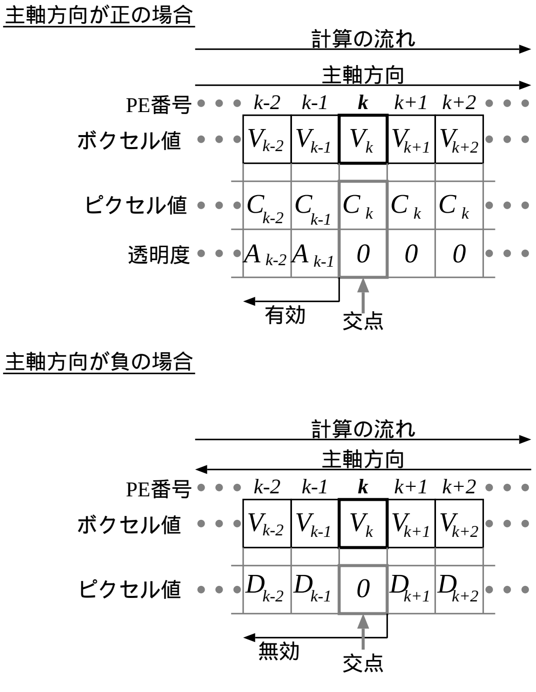


図 21: RT モードでの PCS の計算の様子

3.5.3 PCS の構成

PCS は以下の要素により構成されており、図 22に示すように、PE は 1 次元に接続されている。各 PE は図 4での 1 メモリバンク分のデータについての処理を担当するような構成としている。

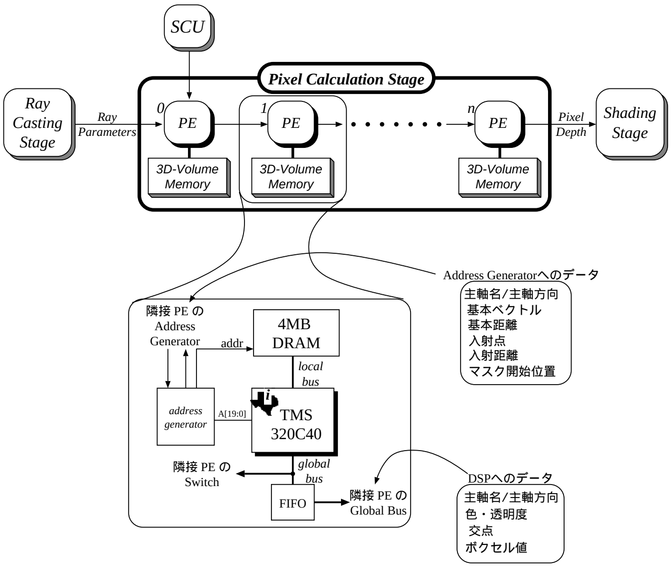


図 22: PCS の構成

- **PE:** 以下の 3 要素により構成されている。
  1. **DSP**$\cdots$ ピクセル値計算、交差判定を行うための DSP。
  2. **Address Generator**$\cdots$ サンプリングするボクセルのアドレスの生成を行う。
  3. **FIFO**$\cdots$ 隣接 DSP 間の通信のためのバッファ。

- **3D-Volume Memory:** 2.2.2節で述べた方法により、ボリュームデータを分割して格納しているメモリ。

これらの要素により PCS では 2 種類のパイプライン処理を行っている。以下では、それぞれの処理方法について述べる。

PE 内パイプライン処理

PCS の PE 内では、DSP、Address Generator、3D-Volume Memory により以下のようなパイプライン処理を行っており、メモリアクセス時間の隠蔽を図っている。

i. 前段 PE からの視線データを Address Generator により受信し、サンプリングするボクセルのアドレスの計算を行う。
ii. Address Generator で計算したアドレスをもとにして 3D-Volume Memory からデータを読み出す。
iii. 3D-Volume Memory からの出力を DSP が取り込み、3.5.1節で述べた方法により色、透明度に変換し、隣接 DSP の結果からピクセル値計算、交差判定を行い、次段の PE へ送信する。

PE 間パイプライン処理

PCS の処理で PE 間のデータ流は大きく 2 つに分けることができる。すなわち、(A)DSP 間を流れるピクセル値計算の途中結果と、(B)Address Generator 間を流れる視線データである。以下では、(A)(B) のデータと、これに関連する処理および構成について述べる。

(A) DSP 間を流れるデータ

i 番目の DSP が送受信するデータの定義、使用目的、大きさについては以下のようなものである。

色 $\cdots$ 式 (7) の C_i または式 (8) の D_{511-i} である。R, G, B それぞれが $16bit$ $float$ で表されている。このデータは通常モード、RT モードに共通である。

透明度 $\cdots$ 式 (7) の A_i である。ただし、視線ベクトルの主軸成分が正の時のみ必要。$16bit$ $float$ により表されている。このデータは通常モード、RT モードに共通である。

交点 $\cdots$ 視線と表面と交差する時の、交点を含んでいる平面の番号。SS で使用される。このデータは、通常モードは $9bit$ であり、RT モードでは平行視線を含めるため、$45bit$ となる。

ボクセル値 $\cdots$ 前段の DSP が読み出したボクセル値。属性テーブルの参照時の差分 (3.5.1節参照) や、交差判定に使用される。ボクセル値は、$8bit$ で表されているため、通常モードは $8bit$ であるが、RT モードでは平行視線の分を含めて $40bit$ になる。

主軸方向 $\cdots$ ピクセル値計算の方法を表す。この正/負により DSP で評価する式 (7) と式 (8) の選択をする。符号のみであるから $1bit$ あればよい。このデータは、通常モード、RT モードに共通である。

DSP 間の通信は単方向であるので、バッファとして FIFO メモリを使用し、DSP の Global Bus に接続する構成としている。一方、DSP の Local Bus には 3D-Volume Memory を接続する構成を採っている。また、これらのメモリは図23のように DSP のメモリマップを使用している。特に、3D-Volume Memory については、 RCS で生成される主軸名のコード (表 2の $AXIS$) がベースアドレスになるように割り付けられなければならない。

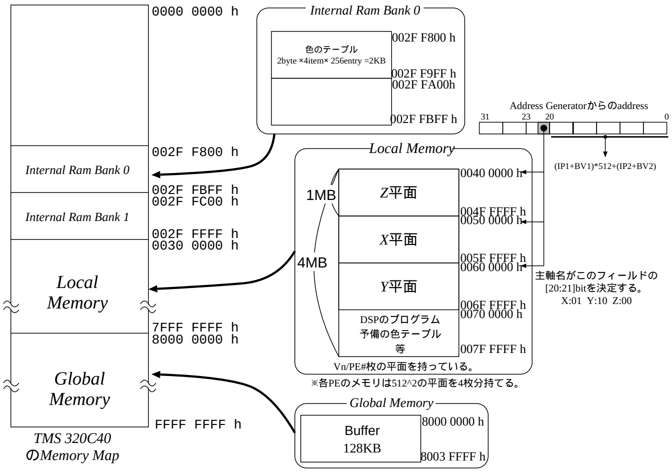


図 23: DSP のメモリマップ

(B) Address Generator 間を流れるデータ

Address Generator 間を流れるデータは、RCS で作成された視線データ (3.4.1節参照) であるが、視線データの内で、入射点のみが PE 間を受け渡される際に以下の式 (9) のような変更を受ける。

$$\text{次段への入射点} = \text{前段からの入射点} + \text{基本ベクトル} \quad (9)$$

以下では、Address Generator における処理ついて述べ、詳しい構成については 4.2.3節で述べる。

(a) アドレス生成$\cdots$ 視線データからサンプリングするボクセルのアドレスを計算する。生成するアドレスは、表 2で示した視線データの記号名を使用して記述すると式 (10) のようになる。

$$AXIS << 20 + (IN1 + BV1) \times 512 + (IN2 + BV2) \quad (10)$$

このうち、$(IN1+BV1, IN2+BV2)$ が式 (9) の右辺に相当し、次段へ送信する入射点となる。ここで、$(IN1+BV1)$、$(IN2+BV2)$ の結果で、片方でも 0 ~ 511 でない時は、サンプリング点がボリューム空間の外にあることになるので、DSP でのピクセル値計算に影響のないように無色・透明のボクセルのアドレスを出力する。

(b) マスクチェック$\cdots$ 視点がボリューム空間の中にある時や、断面の表示では読み出すボクセルが不要かどうかを調べなければならない。この機能は、視線データのマスク開始位置を PE 番号と比較し、主軸方向が正で PE 番号の方が小さい場合、または、主軸方向が負で PE 番号の方が大きな場合はピクセル値計算が不要なので、DSP のピクセル値計算に影響のないように、無色・透明のボクセルのアドレスを出力することで実現されている。

(c) Local Bus 制御$\cdots$ ボクセル値が $1byte$ であるのに対して、DSP のメモリインタフェースは $4byte$ 幅を持つので、

DSP がサンプリングするボクセルのワード内アドレス計算を行う必要があり、不必要な計算が増加する。そこで、DSP の Local Bus を図 24のような構成として、Address Generator の生成するアドレスの下位 $2bit$ により制御することで DSP の byte アクセスの効率化を図る。これにより、DSP では LUB0 命令のみで目的のボクセルをアクセスすることができる。

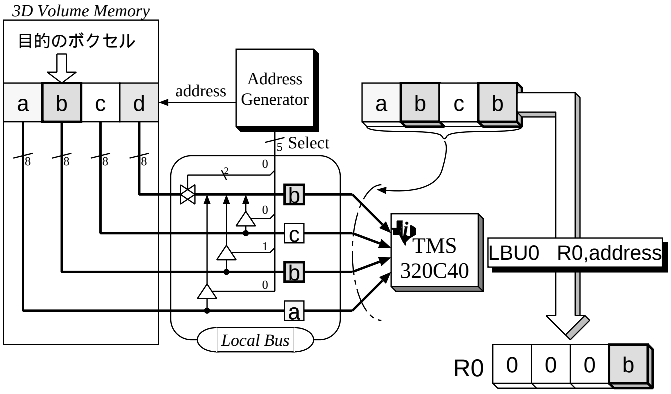


図 24: DSP の Local Bus の構成

ここで、Address Generator が生成する Local Bus の制御信号 (Select 信号) とアドレス、DSP の $R/\bar{W}$の関係を表 3に示す。

表 3: Address Generator からの Select 信号

address の下位 2bit		Select[4:1]	Select[0]‡
Read 時			
0	0	0 0 0 1	1
0	1	0 0 1 0	1
1	0	0 1 0 0	1
1	1	1 0 0 0	1
Write 時			
ϕ	ϕ	0 0 0 1	0

‡:最下位 byte の方向の制御

3.6 シェーディングステージ (SS)

SS は、PCS の計算結果を受け取りフレームバッファへ格納し、SCU からの光源情報に基づいてシェーディング処理を施した後に CRT へ出力するステージである。本節では、通常モードと RT モードの SS の処理の詳細について述べた後、SS の構成について述べる。

3.6.1 通常モードの処理

通常モードにおける処理を大別すると以下に示す 3 種類になる。

・**PCS からの受信**

PCS から与えられるデータで SS の処理に必要なものは、交点、入射距離、基本距離、色である。SS では、これらの受信時に SS で行う計算に必要な形式への変換を行う。まず、交点、入射距離、基本距離を用いて、視点から視線と表面の交差する点までの距離を式 (11) により計算する。

$$depth = \text{入射距離} + \text{交点} \times \text{基本距離} \tag{11}$$

次に、PCS で行っているピクセル値計算では色 (R, G, B) を $16bit\ float$ で計算しているが、SS では $8bit$ の整数として扱うため、PCS からの受信時に型変換を行う。

・シェーディング処理

シェーディング処理では、視線と表面が交差する点での表面の法線を求めることで光源に対して表面がどのような向きにあるかを計算し陰影を付ける。このため法線が必要不可欠であるが、ボクセルは離散的な座標により表されているため、法線は近似により求める以外に方法はない。

$ReVolver$/C40 が採用しているシェーディング方法は depth gradient shading である。このシェーディング方法での法線の計算方法は、各ピクセル毎に Z 値を用意し、スクリーンの水平・垂直方向に隣接する 7 ピクセルの Z 値を用いて計算する方法である (図 25参照)。この方法は、画像生成処理と独立してシェーディング処理を行えるため、パイプライン化により高速化可能である。

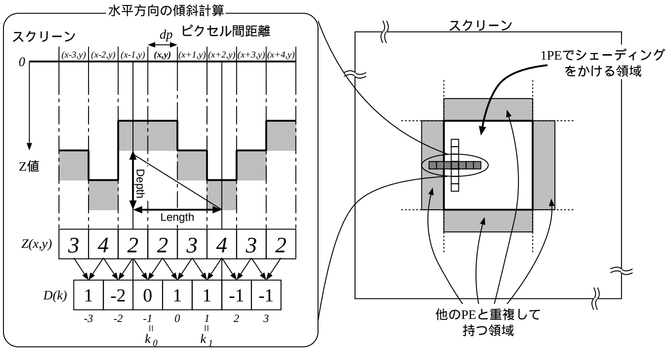


図 25: depth gradient shading

以下では、水平方向のピクセルの傾斜の計算方法を示す。ただし、視点からスクリーンの距離を 1 とした時のピクセルの間隔を dp、ピクセル (x, y) の $Z(x, y)$ で表すとする。

1. $k = 0, \pm1, \pm2, \pm3$ について、$D(k) = Z(x + k, y) - Z(x + k + 1, y)$ を計算する。
2. $\{D(k)|k = 0, \pm1, \pm2, \pm3\}$ で $D(0)$ を含み符号が異ならない最大の範囲を決定する。この範囲内での kの最大値、最小値を k_1、k_0 とする。
3. $Depth = Z(k_1, y) - Z(k_0, y)$、$Length = |k_1 + 1 - k_0|$ を求める。
4. 傾斜角 $\theta_x = \arctan\left(\dfrac{Depth}{dp \times Length \times Z(x, y)}\right)$

同様の処理をスクリーンの垂直方向でも行うことにより、ピクセル (x, y) の法線が求められる。

・出力処理

シェーディング処理を施した画像に対して、必要に応じて特定領域の選択や拡大・縮小の処理を行う。最終的に得られた画像を Frame Buffer へ送出し、CRT へ表示する。

3.6.2 RT モードの処理

RT モードには depth gradient shading は行わず、高次光の方向計算を行うが、高次光の方向を求めるためには

シェーディング処理同様、視線と表面の交点での法線が必要となる。しかし、3.6.1節で述べたシェーディング処理における法線計算ではZ値のみで計算するため、高速であるが誤差が出やすい。また、RTモードには法線により高次光の方向計算がなされるため、より正確な法線が要求される。そこで、$ReVolver$/C40では、3.2.3節で述べた平行視線を用いた方法により、PCSが法線計算の前処理を行う方式を採用している。

以下では、SSがRTモードに行う処理について述べる。

(a) 視線、平行視線の交点のボリューム内座標の計算:PCSから受信する、交点、入射点、基本ベクトルを用いて、図15で示した点A,W,E,N,Sそれぞれのボリューム内座標を計算する。ここで、式(4)の法線計算に用いられるのは$\overrightarrow{AX}$(XはW, E, N, Sのいずれか)であるので、必要なのはAからの相対変位のみである。したがって、法線計算時はA,W,E,N,Sに関して以下の式で求めた座標を使用すればよい。ただし、点Aは高次光の始点となるため、RCSへ送信する際に、正しいボリューム内座標とするため、主軸以外の成分に入射点を加える必要がある。

$$\begin{aligned}\text{ボリューム内座標の主軸成分} &= \text{交点}\\ \text{ボリューム内座標の主軸以外の成分} &= \text{交点}\times\text{基本ベクトル}\end{aligned}$$

(b) 法線計算:(a)の結果をもとにして、式(4)を評価し法線ベクトルを求める。

(c) 高次光の方向計算:(b)で求めた法線ベクトルを正規化したものを$\vec{N'}$で表し、視線ベクトルの単位ベクトルを$\vec{R'}$で表すと、高次光の方向は次の式(12)により求めることができる。

$$\begin{cases}\text{反射光} &= 2\vec{N'} + \dfrac{\vec{R'}}{|\vec{R'}\cdot\vec{N'}|}\\[2ex] \text{屈折光} &= k_f\left(\vec{N'} + \dfrac{\vec{R'}}{|\vec{R'}\cdot\vec{N'}|}\right) - \vec{N'}\end{cases} \tag{12}$$

ただし、式中のk_fは入射側、透過側の屈折率をn_1、n_2で表すと以下のようになる。なお、屈折率は、PCSでの属性テーブルと同様のテーブルを、あらかじめSSの全PEに用意しておき、PCSから与えられるボクセル値をインデックスとして参照することにより求める。

$$k_f = \left(\left(\frac{n_1}{n_2}\right)^2\left|\frac{\vec{R'}}{\vec{R'}\cdot\vec{N'}}\right|^2 - \left|\frac{\vec{R'}}{\vec{R'}\cdot\vec{N'}} + \vec{N'}\right|\right)^{-\frac{1}{2}}$$

(d) ピクセル値の書き込み:PCSから送れらてくる 色と該当するフレームバッファのピクセルの和を、Frame Bufferへ書き込む。

(e) 画像生成の終了判定:画像生成は、全ピクセルに関して反射屈折が一定の回数だけ起こるか、高次光を発生させない状況になれば終了する。そこで、反射屈折のための視線を生成するSSによりピクセル毎に高次光の生成回数をカウントすることで画像生成の終了判定が行われる。

3.6.3 SSの構成

3.6.1、3.6.2節の処理を行うためのSSは、以下の要素から構成されており、図26に示すようにメッシュ状に結合されている。

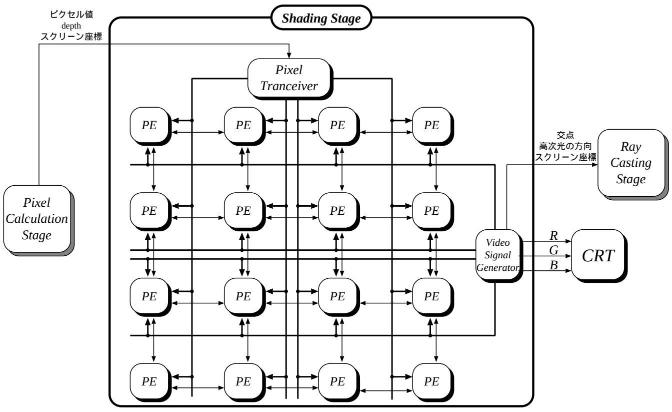


図 26: SSの構成

- **<u>PE</u>:** 以下の要素から構成されている。
  - **DSP:** シェーディング処理/高次光の方向計算を行うためのDSP。
  - **Receive Buffer:** PCSからのデータの受信用のバッファ。
  - **Output Buffer:** Video Signal Generatorへの出力用のバッファ。VRAMを使用する。
- **<u>Pixel Tranceiver</u>:** スクリーン座標から適切なPEへピクセル値などのデータを分配する。
- **<u>Video Signal Generator</u>:** CRTの同期信号、CRTへの画像出力、RCSへの高次光の送信の中継を行う。

通常モードの処理では、PCSからの受信、シェーディング処理、画像出力を同時に行うため、Recive Buffer、Output Bufferはそれぞれダブルバッファリングにより、処理が停滞しない構成となっている。また、シェーディング処理ではスクリーンをブロック分割し、サブスクリーン毎にPEを割り当てる方式で並列処理を行っている。しかし、単純なブロック分割ではシェーディング処理の際にPE間通信が発生するため、図25のようにデータを隣接PEと重複してデータを持つことでPE間通信を避けるようにしている。ここで、通常モードの処理に必要な命令数を調べたところ、151命令であることが分かった。仮に、リアルタイム($33.3ms/frame$)で処理を行うとすると、SSに必要なPE数は約23.8となり、比較的少数である。

RTモードでは、スパンコヒーレンスのため、高次光を発生させるピクセルは連続している可能性が高い。そこで、PE間の負荷分散を図るためには、スクリーンをできるだけ細かく分割してPEをサイクリックに割り当てて処理を行う方法が考えられる。しかし、Ouptput BufferとしてVRAMを使用しておりSAMポートの高速読み出しが128個のデータを単位としているため、$ReVolver$/C40では、スクリーンを$128 \times 1pixel$に分割しPEをサイクリックに割り付けるようにしている。

また、RTモードでの1ピクセル当たりの処理量について調べたところ、5つの交点のボリューム内座標計算、式

(4) の法線計算、式 (12) の高次光の方向計算に要する命令のの合計が 392 命令であった。残りのピクセル値の書き込み、画像生成の終了判定を含めても 500 命令あれば充分に処理が完了すると考えられる。仮に、RT モードでリアルタイム処理を行うとしても、約 78.8PE 程度の構成となる。しかし、RT モードでは PCS の処理速度も低下するので、実際に必要な PE 数はこの数よりも小さくなる。この問題については、4.3節で詳しく述べる。

4　プロトタイプ構成

本節では、現在開発しているプロトタイプでの拡張機能と、その構成について述べた後、最後にプロトタイプにおける性能について述べる。

4.1　プロトタイプでの拡張機能

本研究では *ReVolver*/C40 のアーキテクチャの有効性を確認するために、小規模なプロトタイプの設計、開発を行っている。これは、表 4に示すように、152PE から成るシステムである。しかし、PCS が 128PE なので $512^3 voxel$ の空間をレンダリングするには PE 数が不足している。そこで、PCS の PE に変更を加えて図 27に示すように、1PE で複数の平面の処理を行えるようにし、小規模な構成でも大きな空間の可視化を行うようにする。

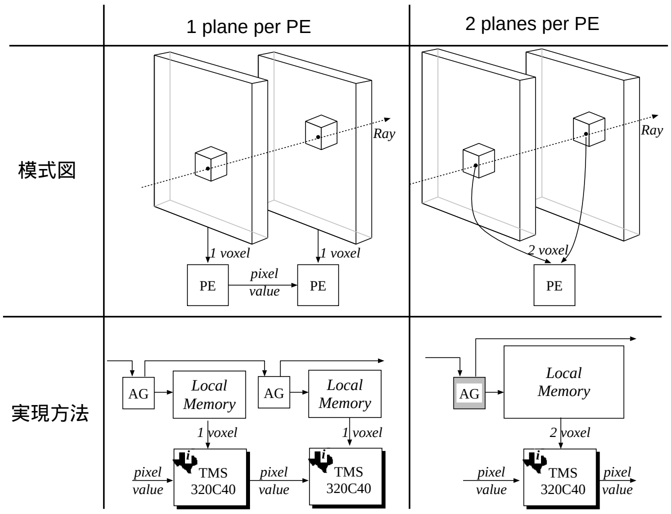


図 27: PCS の PE による複数平面の担当

この変更は、ボリュームデータを格納するメモリに関しては、単に容量を平面の数に応じて増量するだけでよいが、そのメモリからサンプリングするボクセルのアドレスを生成する Address Generator に関しては、アドレス生成、マスクチェックを複数の平面に対して行うような機構としなければならない。この Address Generator に関する構成については、4.2.3節で詳しく述べる。

4.2　プロトタイプの構成

4.2.1　規模の設定

ReVolver/C40 はパイプライン計算機であるため、各ステージ間での処理時間のバランスがとれるように PE を配分しなければならない。そこで、本節では各ステージでの処理時間を見積もることで、それぞれのステージに必要な PE 数を求める。なお、RCS、SS の PE 数の計算では、スクリーンのサイズを $512^2 pixel$ として計算する。

・RCS の PE 数

RCS-PCS 間でバランスを保つためには、PCS 内のパイプラインピッチで、RCS から PCS へ視線データを渡せればよい。PCS が最も高速なのは4.3節で述べるように不透明ボリュームの表示を行う際で、PCS 内のパイプラインピッチは $260ns$ となる。RCS では 1 視線を生成するのに 67 命令必要なことから、RCS の PE は $(67\times 20ns)/260ns =$ 約 5.2 個程度必要であると計算できる。よって、RCS の PE 数には 8 個の PE を配置すれば十分である。

・PCS の PE 数

PCS は 1 次元のパイプライン構造で、その PE 数は、ボリューム空間の 1 辺の大きさのみで決められる。*ReVolver*/C40 では、他のステージを PCS の性能に合わせることで全体のバランスを保つようにしている。

今回のプロトタイプでは、PCS に 128PE を用意し、各 PE に $512^2 \times 4voxel$ を格納できるようにする。すなわち、各 PE が最大 4 平面を担当することで、$512^3 voxel$ のボリュームデータをレンダリングを行えるようにする。

・SS の PE 数

SS の処理と、RCS と PCS の処理はスクリーンを単位としたパイプラインであり、RCS、PCS で 1 フレームを生成する間に、SS では 1 フレームの depth gradient shading が行えればよい。RCS、PCS による画像生成が最速であるのは不透明ボリュームの表示の時の約 $68ms/frame$ である。SS で $1pixel$ 分の depth gradient shading 処理に 151 命令必要であることから、SS には $(151\times 20ns)/68ms =$ 約 11.6 個程度の PE が必要となる。したがって、SS には 16 個の PE を配置することにする。

以上をまとめると、*ReVolver*/C40 のプロトタイプの規模は表 4のようになる。

表 4: プロトタイプの規模

・ハードウエア規模	
RCS	8PE
PCS	128PE
SS	16PE
計	152PE
・データの規模	
ボリュームデータ	最大 $512^3 voxel$
スクリーン	最大 $512^2 pixel$

4.2.2　RCS の実装

本節では、*ReVolver*/C40 のプロトタイプにおける、RCS の詳細な構成について述べる。

図 28に RCS の View Buffer、PE、Ray Buffer の構成を示す。View Buffer へのアクセスには、排他制御が必要なため、PE にはそれぞれ **GSM-IF**(Global Shared Memory InterFace) を配置し、View Buffer へのアクセス要求を **GBC**(Global Bus Controller) へ出す。**GBC** では PE の優先度にしたがって View Buffer へアクセスできる PE を決定するような構成となっている。

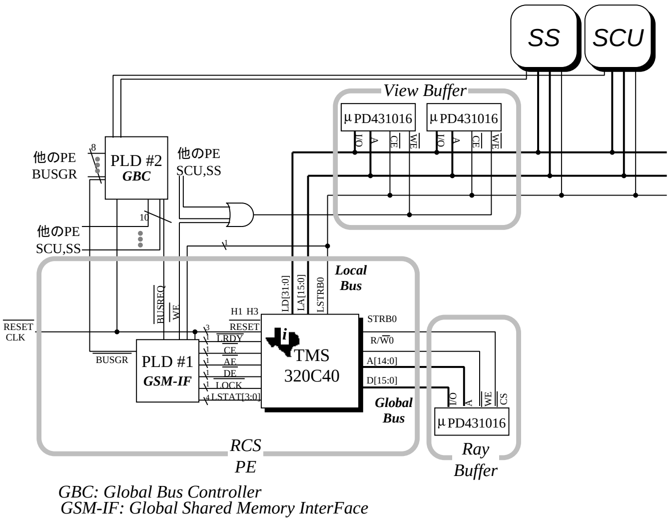


図 28: RCS の PE、View Buffer、Ray Buffer の構成

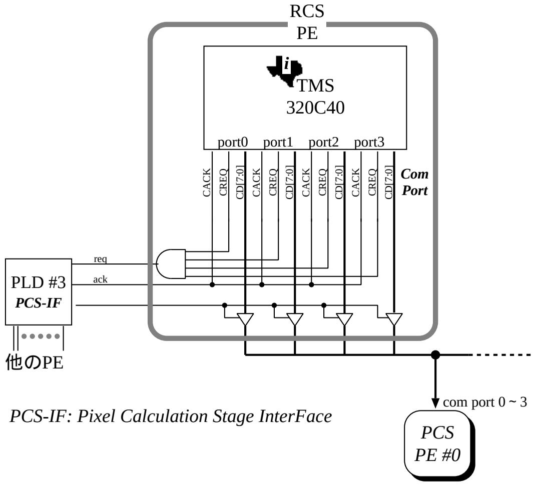


図 29: RCS の PCS のインタフェース

RCS と PCS のインタフェースは、図 29に示すような構成となっており、各 PE の DSP の com port を PCS の 0 番の PE の com port に直結した共有バス構成となっている。このバスの制御は **PCF-IF**(PCS InterFace) により行われる。

ここで、View Buffer ↔ PE 間と、PE ↔ PCS 間の共有バスについて、各構成要素からのアクセス頻度から必要なバンド幅を求める。

まず、View Buffer へのアクセスについては、SCU、SS、RCS の PE からによるものに分けられ、まとめると以下のようになる。

SCU: 1 フレームの生成時間 (約 $68ms$) に 1 回、視点/スクリーンのパラメータ ($30byte$) の更新を行うため、最大でも $0.44Kbyte/sec$ となる。

RCS の PE: 1 フレームの生成時間に 1 回、全 PE が、視点/スクリーンのパラメータの獲得を行うため、最大でも $3.45Kbyte/sec$ となる。

SS: RT モードにおける高次光の計算時間 (約 500 命令 × $20ns$) に 1 回、高次光のデータ ($22byte$) を書き込む。SS には 16PE あるので、約 $33.6Mbyte/sec$ となる。

よって、View Buffer に要求されるバンド幅は最大でも $33.6Mbyte/sec$ 強と DSP の Local Bus のバンド幅 ($100Mbyte/sec$) に比べて小さく、バス競合による PE の並列動作の障害はないものと考えられる。

つぎに、RCS の PE から PCS への PE へのデータ送信については、視線データは $18byte$ で表されるので、PCS の最速のパイプラインピッチ ($260ns$) で送出すると、約 $66Mbyte/sec$ のバンド幅が要求される。そこで、RCS と PCS 間のリンクは、$20Mbyte/sec$ のバンド幅を持つ DSP の com port を 4 本使用して構成される。

4.2.3 PCS の実装

·Local/Global Bus の構成

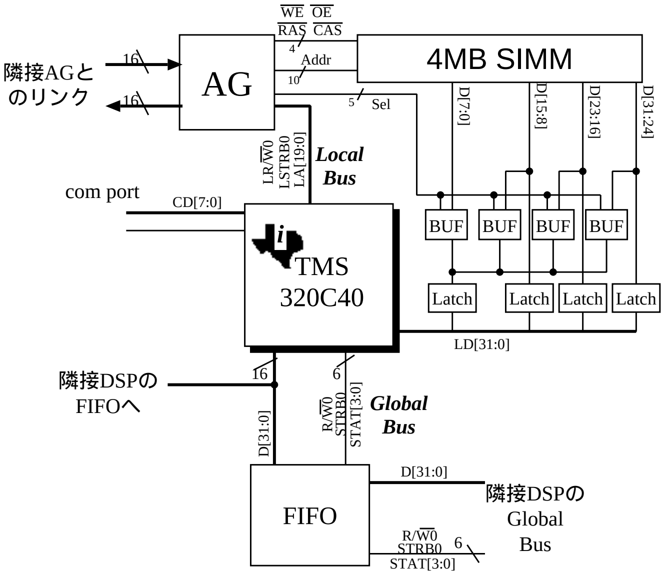


図 30: PCS の PE の構成

PCS の詳細な構成を図 30に示す。DSP の Local Bus、Global Bus にはそれぞれ $4Mbyte$DRAM、FIFO を配置し、それぞれ Volume Memory、隣接 PE 間通信のバッファとして使用している。Local Bus では、3.5.3節の図 24で示した DSP の $byte$ アクセスを効率化するための構成に使用するバッファと、DSP と Address Generator+Memory のタイミングを合わせるための Latch を配置している。

Global Bus に配置している、通信用のバッファの FIFO は $10ns$ のものを使用しており、十分に DSP で 0wait アクセスを実現できる。

·Address Generator の構成

3.5.3節で述べた Address Generator は、図 31に示す通り、以下の構成要素から成っている。なお、以下では $512^3 voxel$を 128PE で処理する場合の構成、すなわち各 PE が 4 平面を担当する場合について述べる。

– 加算器 (2 つ)

式 (10) の入射点と基本ベクトルの加算を行うための $12bit$ 加算器である。入力データは、入射点のデータ、基本ベクトルであり、加算器の出力は Address Latch でアドレスの一部として使用される。ただし、加算器の出力で $[10:9]bit$ 目に 1 があれば、サンプリング点がボリューム空間の外で

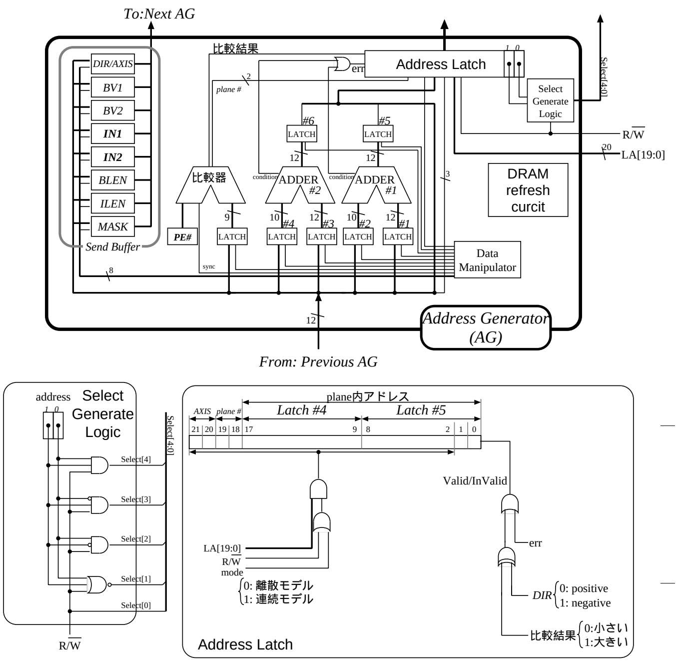


図 31: Address Generator の構成

あることを示しているので、Address Latch へ入力している condition を 1(エラー) とする。

また、加算器の出力は次段の Address Generator へ送信する入射点ともなるため、Send Buffer の $IN1$、$IN2$ へ格納される。同時に、1PE で 4 平面の処理を行うため、加算器の出力はもう一度加算器の入力 Latch にも入れられる。

– 比較器

PE 番号とマスク開始位置を比較するための $9bit$ 比較器である。この比較器では、1PE が 4 平面を担当しているため、単に PE 番号とマスク開始位置と比較するのではなく、PE 番号から (PE 番号+4) までの番号と比較しなければならない。

そこで、マスク開始位置の下位 $2bit$(担当している平面数の対数分の bit 数、この値を m とする) を無視した値と PE 番号と比較し、PE 番号の大/小に応じて 1/0 を 4 回出力する。PE 番号と同じ場合は、0 を m 回、1 を $4-m$ 回出力する。ただし、Address Latch への出力は加算器によるアドレス計算と同期する必要があるため、Data Manipulator からの信号により出力を行う。

さらに、比較結果の出力が出された回数もボクセルのアドレスとして必要であるため、$2bit$ カウンタを設置し、比較器からの出力毎にカウントを行い、Address Latch へ出力する。

– **Address Latch**

512^3voxelを 128PE で分割して保持するため、$ReVolver$/C40 では各 PE に、4MB のメモリを配置している。よって、サンプリングされるボクセルのアドレスは $22bit$ となる。このアドレスを保持するための Latch は以下のようなフィールドにより構成されている。

- AXIS $\cdots$ RCS で生成された主軸のコード (表 2の $AXIS$) を [21:20]bit 目に格納する。
- plane# $\cdots$ 比較器のカウンタの出力を [19:18]bit 目に格納する。
- plane 内アドレス $\cdots$ 加算器の結果をそれぞれ [17:9]bit 目、[8:0]bit 目に格納する。

また、Address Latch では、主軸データと同時に送られてくる主軸方向と比較器から与えられる比較結果の排他的論理和の結果と加算器からの condition の論理和により、生成したアドレスが有効か無効かを決定する。無効な場合は、無色 ·透明のボクセルのアドレスを出力するため、0x0070 0000 ~ 0x007F FFFF までの特定の値にリセットにする機構を持つ。

– **Seletct** 生成回路

Local Bus を制御するための信号 (表 3) を生成するための回路。入力は、Address Latch の下位 $2bit$ と DSP の $R/\bar{W}$である。

– その他

入力データを Address Generator 内部の適切な場所へ格納するための制御信号を生成する Data Manipulator と、次段の Address Generator へデータを送信するための Send Buffer、および、DRAM refresh 回路を内蔵している。

4.3 性能見積り

本節では、プロトタイプでの性能の見積りを行う。ここで、性能見積に関しては、ボリュームの大きさは、128^3voxelであり、スクリーンの大きさは 512^2pixelであるとしている。また、使用する DRAM はランダムアクセスで、110ns であるとする。

4.3.1 性能の見積り方法

$ReVolver$/C40 では、PCS に合わせて他のステージの PE 数が決まっているため、全体性能は PCS のみに依存する。したがって、性能の見積りは、PCS の性能を各モード毎に処理時間を求めることにより行えばよい。そこで、各モードの処理を行うプログラムを作成し、必要な命令数から処理時間を計算することにする。まず、PCS の処理を大別すると、以下のように分類することができる。

1. Address Generator によるアドレス生成時間
2. DRAM からのボクセル値の読み出し時間
3. ピクセル値の計算時間
4. データの送受信時間

これらの時間のうち、1のアドレス生成必要な時間は、主に加算器に必要な時間のみであるので、2の DRAM アクセスや 3のピクセル値計算で必要な時間に比べて小さくパイプライン処理を乱す要因とはならないと考えられる。また、4のデータの送受信時間については、DSP の命令として実現されている。したがって、性能の見積りで考慮が必要なのは、DRAM のアクセス時間と、DSP の処理時間ということになる。

ここで、2の時間は、読み出すボクセルの個数に応じて変化するため、各モードでのボクセルのアクセスパターンを図 32にまとめる。

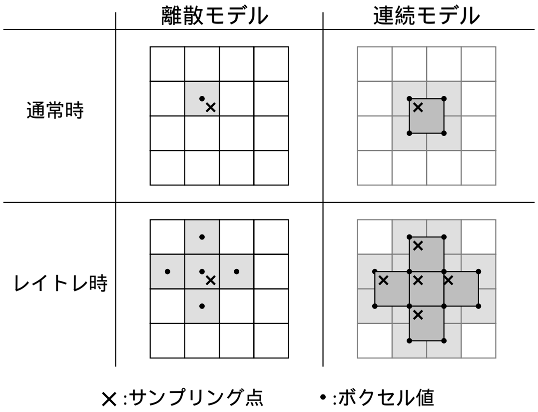
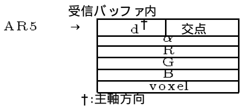


図 32: 各モードでのボクセルのアクセスパターン



†:主軸方向

Address Register(AR) の使用方法		
AR1	→	属性テーブル (R) の先頭
AR2	→	属性テーブル (G) の先頭
AR3	→	属性テーブル (B) の先頭
AR4	→	属性テーブル (α) の先頭
AR5	→	send/receive 時アドレス
AR6	→	1(0),tmp(±1)
AR7	→	R(0),G(1),B(-1)

() 内の数字は displacement

```
LOOP:AND        *+AR5(0),@_FFFF; 主軸方向のチェック
     BND        NEG            ; if(DIR==1) brach
     LDF        R5,*+AR5(0)    ; R5 ← α
  || LDF        R7,*+AR6(0)    ; R7 ← 1
     LBU0       @_volume,IR0   ;ボクセル値のロード
                          ; 主軸が正の時の処理
     MPYF3      R5,*+AR1(IR0),R0; R0 ← R(v_i) × α
  || SUBF3      *+AR6(0),R7,R2 ; R2 ← (1 − α(v_i))
     MPYF3      R5,*+AR2(IR0),R1; R1 ← G(v_i) × α
  || STF        R0,*+AR6(1)    ; tmp ← R0
     MPYF3      R2,*+AR6(1),R0 ; R0 ← R(v_i) × α × (1 − α(v_i))
  || STF        R1,*-AR6(1)    ; tmp ← R1
     MPYF3      R2,*-AR6(1),R1 ; R1 ← G(v_i) × α × (1 − α(v_i))
  || ADDF3      R0,*+AR5(0),R3 ; R3 ← R(v_i) × α × (1 − α(v_i)) + R
     ADDF3      R1,*+AR5(0),R1 ; R1 ← G(v_i) × α × (1 − α(v_i)) + G
  || STF        R3,*+AR7(0)
     MPYF3      R5,*+AR3(IR0),R0; R0 ← B(v_i) × α
  || STF        R1,*+AR7(1)
     MPYF3      R5,*+AR4(IR0),R1; R1 ←α × α(v_i)
  || STF        R0,*+AR6(0)    ; tmp ← R0
     MPYF3      R2,*+AR(0),R0  ; R0 ← B(v_i) × α × (1 − α(v_i))
     ADDF3      R0,*+AR5(IR0),R0; R0 ← B(v_i) × α × (1 − α(v_i)) + B
     BRD        LOOP
     LDF        *+AR7(1),R6
  || LDF        *+AR7(0),R7
     STF        R1,*+AR5(0)    ; send R1(new α)
  || STF        R7,*+AR5(0)    ; send R7(new R)
     STF        R6,*+AR5(0)    ; send R6(new G)
  || STF        R0,*+AR5(0)    ; send R0(new B)
     STF        IR0,*+AR5(0)   ; send IR0(new voxel)
                          ; 主軸が負の時の処理
 NEG: MPYF3     R5,*+AR5(0),R0 ; R0 ←α(v_i) × R
  || SUBF3      *+AR4(IR0),R7,R2; R2 ← (1 − α(v_i))
     MPYF3      R2,*+AR1(IR0),R1; R1 ← (1 − α(v_i)) × R(v_i)
  || STF        R0,*+AR7(0)
     MPYF3      R5,*+AR5(0),R0 ; R0 ←α(v_i) × G
  || ADDF3      *+AR7(0),R1,R3 ; R3 ← (1 − α(v_i)) × R(v_i) + α(v_i) × R
     STF        R5,*+AR5(0)    ; send R1(new α) for dummy
  || STF        R3,*+AR5(0)    ; send R3(new R)
     MPYF3      R2,*+AR2(IR0),R1; R1 ← (1 − α(v_i)) × G(v_i)
  || STF        R0,*+AR7(1)
     MPYF3      R5,*+AR5(0),R0 ; R0 ←α(v_i) × B
  || ADDF3      *+AR7(1),R1,R3 ; R3 ← (1 − α(v_i)) × G(v_i) + α(v_i) × G
     MPYF3      R2,*+AR3(IR0),R1; R1 ← (1 − α(v_i)) × B(v_i)
     BRD        LOOP
  || STF        R0,*-AR7(1)
     ADDF3      *-AR7(1),R1,R0 ; R0 ← (1 − α(v_i)) × B(v_i) + α(v_i) × B
  || STF        R3,*+AR5(0)    ; send R3(new G)
     STF        R0,*+AR5(0)    ; send R3(new B)
  || STF        IR0,*+AR5(0)   ; send R3(new voxel)
```

図 33: ボリュームレンダリング処理のプログラム (半透明処理)

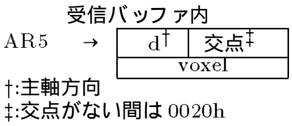
4.3.2 通常モードの性能

通常モードの離散モデルのボクセルのレンダリングでは、図 32によりアクセスするボクセルは 1 つであるので、DRAM からの読み出し時間は $110ns$ と考えてよい。次に、DSP では図 33に示したプログラムを実行する。このプログラムの最長経路は、主軸が正の場合で、18 命令必要となっている。よって、DSP の処理時間は $18 \times 20ns$ となる。したがって、DRAM のアクセス時間より DSP の処理時間の方が長く、PE 間のパイプラインでは DSP による処理時間が支配的となる。したがって、このモードでの性能は、$360ns \times 512^2 = 94.37ms/frame$ となる。

通常モードのオプション使用時の性能

- 不透明ボリュームのレンダリングを行う場合:この際の PCS の処理は、表面と交差する際のボクセル値のみを SS へ送信し、SS のシェーディング時に色へ変換されるようにするため、PCS ではボクセル値のみが DSP 間で受け渡すことになる。

半透明ボリュームのレンダリング同様、アクセスするボクセルは 1 つであり、図 34の不透明ボリュームのレンダリングを行うプログラムの最長経路は、主軸が正の場合に交点のボクセルを処理する際の 13 命令 ($260ns$) である。したがって、この場合も DSP による処理時間が支配的となるので、$260ns \times 512^2 = 68.1ms/frame$ という性能になる。



初期値	
R10	(PE 番号)
AR6	8000h(1),0020h(-1)

†:主軸方向
‡:交点がない間は 0020h

```
   LOOP:
                  LDI      *+AR5(0),R0        ; 受信
         ||       LDI      *+AR5(0),R1        ; 受信
                  TSTB     R0,*+AR6(1)        ;
                  BND      NEGATIVE           ;
                  LUB0     R2,@_volume        ;
                  LDINZ    R2,R3              ;
                  NOP                         ;
                  BRD      SEND               ;
                  TSTB     R0,*-AR6(1)        ;
                  LDINZ    R10,R0             ;
                  LDINZ    R3,R1              ;
NEGATIVE:
                  AND      R2,R2              ;
                  LDINZ    R10,R0             ;
                  OR3      R0,*+AR6(1),R0     ;
   SEND:
                  BRD      LOOP               ;
                  STI      R0,*+AR5(0)        ;
         ||       STI      R1,*+AR5(0)        ;
                  NOP                         ;
                  NOP                         ;
```

図 34: ボリュームレンダリング処理のプログラム (不透明時)

- 連続モデルのボクセルを扱う場合:連続モデルでは、図 10の式 (2) を評価しなければならないため、DSP がサンプリング点の座標計算を行わなければならない。したがって、Address Generator へ入力していたデータも DSP へ入力するようにする。

このアドレス計算、tri-linear 補間の計算を含めた処理に必要な命令数は 174 命令†($3.84\mu s$) であり、ボクセルの読み出し時間が $440ns$ であることから、DSP の処理時間がパイプラインピッチとなる。したがって、処理性能は $3.84\mu s \times 512^2 = 0.9sec/frame$ となる。

- サンプリング間隔の補正を行う場合:サンプリング間隔の補正は、ボクセル値から色、透明度に変換する際に 3.2.2 節の式 (3) を評価することにより行われる。この式の評価には 17 命令†($340ns$) 余分に必要なことから、補正を行うと通常モードで、約 $183.5ms/frame$ となる。

4.3.3 RT モードの性能

RT モードにアクセスするボクセル数は図 32に示すように、5 ボクセルである。したがって、ボクセルの読み出し時間は $550ns$ となる。一方、RT モードの処理に必要な命令数は、92 命令† ($1.84\mu s$) である。さらに、SS の 1PE の処理速度は約 500 命令で 1 高次光の処理が行えるので、理想的に負荷が分散されたとすれば、$0.625\mu s$/高次光の処理能力を持つことになる。PCS の性能が $1.84\mu s/pixel$であ

ることから、SS の処理能力は PCS に比べて十分に高く、SS によるパイプラインの乱れはないと考えられる。

したがって、RT モードの性能も PCS の DSP の処理時間で決まる。よって、全ての視線が 2 回反射屈折を繰り返したとすると、処理すべき視線の数が$\sum_{i=0}^{2} 2^i = 7$倍になるので、約 $3.34sec$ で 1 フレームを生成できることになる。

4.3.4　既存の並列計算機との比較

ReVolver/C40 のプロトタイプと既存の並列計算機との比較を行い、本節のまとめとする。比較の対象とするのは専用のハードウェアを用いた計算機と汎用計算機での並列アルゴリズムによるものである。これら概要を表 5 にまとめる [17]。以下では、専用のハードウェアの例として CUBE、汎用機を用いた例として CM-5 を取り上げて述べることにする。

通常モードの比較 — CUBE—

CUBE は *ReVolver*/C40 と同様に後方投影アルゴリズムを採用している専用並列計算機である。しかし、*ReVolver*/C40 と異なり、CUBE が高速なのは、視線方向が ±x,±y,±z の 6 通りのいずれかであり、平行投影を行う場合のみである。

これは、CUBE のメモリ構造が、n をボリューム空間の１辺とした時、(p, q, r) のボクセルを $k = (p+q+r) \bmod n$ で定められる番号のメモリバンクに格納する構成となっているからである。視線が軸と平行にならない場合はボリュームメモリの内容を回転してから、画像生成行うため、表 5 に示すように投影が $62ms$ であるのに対し、回転を伴うと $573ms$ に性能低下する。

これに対して、ReVolver では視線の方向に無関係に不透明時で $68ms$、半透明時で $94.4ms$ で投影が可能であり、視点移動が伴うようなインタラクティブな使用にも問題ない。

RT モードの比較 — CM-5 —

ReVolver/C40 のように、レイトレーシングをサポートしている専用の並列計算機がないため、汎用並列計算機 CM-5 を用いた例を示す。

参考文献 [18] で述べられている方法は、ボリューム空間を幾つかのサブボリュームに分割し、サブボリューム毎にプロセッサが画像生成を行った後、これらの画像をマージして、最終的な画像を得るものである。このアルゴリズムを、CM-5 上で比較的密なボリュームデータのレンダリング結果が表 5である。

しかし、128PE の *ReVolver*/C40 で、全ピクセルで 2 回の反射屈折が起きる場合でも 3.39 秒と 512 台構成の CM-5 と対等な性能であり、高速であることが分かる。

5　おわりに

ボリュームレンダリング専用並列計算機 *ReVolver*/C40 のアーキテクチャとそのプロトタイプの構成について述べた。

ReVolver/C40 では、科学技術計算の結果の可視化に必要な、半透明ボリュームの遠近法による画像生成を高速に行うための機構として、3 重化されて格納されたボリュームデータにより任意方向の視線上のボクセルを読み出せるメモリ構成を採用している。さらに、汎用の DSP を用いて視線生成、ピクセル値計算、シェーディングの各処理を行うステージを構成しており、柔軟な処理を行うことが可能となっている。

また、*ReVolver*/C40 のプロトタイプでは、ボリューム空間の大きさと PCS の PE 数の関係を緩和することにより、さらに柔軟性を増した構成となっている。このプロトタイプは視線生成処理に 8PE、ピクセル値計算に 128PE、シェーディングに 16PE の構成で、最大 $512^3 voxel$までのレンダリングを可能としている。

プロトタイプの性能については、ボリューム空間が $128^3 voxel$では、不透明ボリュームのレンダリングで $68.1ms/frame$、半透明ボリュームでは $94.4ms/frame$ という性能となっている。

今後の課題としては、*ReVolver*/C40 の柔軟性を生かして、*ReVolver*/C40 自身によるボリュームデータの作成や、様々な画像生成手法の実装と評価などが挙げられる。特に前者については、*ReVolver*/C40 の応用範囲を広げるための研究として、非常に重要な研究テーマである。

謝辞

本研究の機会を与えてくださり、適切な御指導を賜わった富田眞治教授に深甚な謝意を表します。

数々の有用な御意見を戴いた中島浩助教授、森眞一郎助手に心から感謝致します。

本研究の共同研究者である、金喜都氏、中山明則氏、荻野 友隆氏をはじめ、計算機システム講座の諸氏に感謝いたします。

参考文献

[1] 對馬 雄次, 明石 英也, 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,JSPP'94, pp89-96, 1994.

[2] 對馬 雄次, 金 喜都, 他:ボリュームレンダリング専用並列計算機 ReVolver/C40 のアーキテクチャ, 情報処理学会研究報告 94-ARC-107, pp137-144, 1994.

[3] 明石 英也: ボリューム ・レンダリング向き並列計算機のアーキテクチャ, 京都大学修士論文,1992-2

[4] S.M. Goldwasser: A Generalized Object Display Processor Architecture, IEEE Compuuter Graphics & Applications, 4, 10, pp. 43-55,Oct. 1984.

[5] 内木 哲也, 所 真理雄: ３次元メモリを用いた立体図形表示機構 - SCOPE -, 電子通信学会論文誌, Vol.J68-D, No. 4, pp. 741-748, 1985.

[6] Toshiki OHASHI, Tetsuya UCHIKI and Mario TOKORO: A Three-Dimensional Shaded Display Method for Voxel-Based Representation, EUROGRAPHICS 85, pp. 221-232, 1985.

[7] D. Jackel and W. Strasser: Reconstructing Solids from Tomographic Scans -The PARCUM II System-, Advances in Computer Graphics Hardware II, pp. 209-227, 1988.

[8] A. Kaufman and R. Bakalash: Memory and Processing Architecture for 3D Based Imagery, IEEE Computer Graphics & Applications, 8, 6, pp. 10-23, Nov. 1988.

[9] M. Levoy: Display of Surfaces from Volume Data, IEEE Computer Graphics & Applications, Vol. 8, No. 5, pp. 29-37, 1988.

[10] P. Sabella: A Rendering Algoritm for Visualizing 3D Scalar Fields, Computer Graphics, Vol. 22, No. 4, pp. 51-58, 1988.

[11] Kevin L.Novins: An Efficient Method for Volume Rendering using Perspective Projection,Computer Graphics Vol 24 No.5,pp 95-102,Novermber 1990

[12] James T. Kajiya, Brain P. von Herzen: RAY TRACING VOLUME DENSITIES, Computer Graphics, Vol. 18, No 3, pp. 165-175, 1984

[†] C 言語を用いて記述したプログラムをコンパイラによりアセンブラに変換し必要な命令を調べたため、多少余分なコードが含まれている。

表 5: *ReVolver*/C40 と既存の計算機との比較

専用のハードウェアの例

名称	並列処理方式	ボリューム	投影法	扱うボリューム (速度)
GODPA	ボリューム並列	256^3	平行投影	不透明 (20 ~ 30 画像/秒)
3DP^4	ボリューム並列	$64^3 \times 256$	遠近法も可	不透明 (10 画面/秒†)
CUBE	ボクセル並列	512^3	遠近法も可	不透明 [半透明も可能] (軸方向の平行投影 62ms 回転を伴う描画 573ms 任意方向の描画 573ms)
ReVolver/C40	{ ボクセル ピクセル } 並列	128^3	遠近法を サポート	不透明 (投影/回転 68.1ms) 半透明 (投影/回転 94.4ms)

† :ソフトウェアシミュレーションによる性能

汎用計算機を用いた例

名称	PE 数	画像生成法	ボリューム (スクリーン)	速度
CM-5	512	レイトレーシング法	$256^3(512^2)$	3.19 秒
	128	レイトレーシング法	$256^3(512^2)$	12.7 秒
DECmpp	16K	レイキャスティング法	$128 \times 128 \times 112(128^2)$	238ms
	1K	レイキャスティング法	$128 \times 128 \times 112(128^2)$	1.39 秒
AP-1000	128	レイキャスティング法	$256 \times 256 \times 109(256^2)$	約 50 秒
ReVolver/C40	128	レイトレーシング法	$128^3(512^3)$	3.39 秒

[13] A. Kaufman: The voxblt Engine: A Voxel Frame Buffer Processor, Advances in Computer Graphics Hardware III, pp. 85-102, 1991.

[14] R. A. Drebin, L. Carpenter and P. Hanrahan: Volume Rendering, Computer Graphics, Vol. 22, No.4, pp. 64-75, 1988.

[15] 田山典男, 清水則明, 漆間文俊: 可変歩幅 DDA による 3 次元メモリ空間での高速なボクセル追跡, 情報処理学会論文誌, Vol. 32 No.5 , pp. 581-589,1991

[16] 荻野 友隆:ボリュームレンダリング専用並列計算機 *ReVolver*/C40 のシェーディングステージ, 京都大学特別研究報告書 1995-2

[17] Arire Kaufman,Reuven Bakalask,Daniel Cohen,and Roni Yagel: Architectures for Volume Rendering,VOLUME VISUALIZATION Chapter 6,pp311-320, 1991

[18] Kwan-Liu Ma, James S. Painter, Charles D.Hansen, Michael F.Krogh: A Data Distributed, Parallel Algorithm for Ray-Traced Volume Rendering,PARALLEL RENDERING SYMPOSIUM, pp.15-22, 1993.

[19] Texas Instruments: TMS 320C4x User's Guide, 1993