ボリュームレンダリング専用並列計算機

—— *ReVolver*/C40 ——

對馬 雄次[‡], 中山 明則, 荻野 友隆, 金 喜都
森 眞一郎, 中島 浩, 富田 眞治
京都大学 工学部 (‡: 現在 日立製作所)

本稿では、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現在開発を進めているプロトタイプについて述べる。*ReVolver*/C40 の特徴は、3 ステージマクロパイプライン構成と各ステージの並列処理による高速化、任意方向の視線に対してバンクコンフリクトしないメモリ構成である。これらの特徴により、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ことを可能としている。また、*ReVolver*/C40 の特徴を生かし、視線方向の定まらないレイトレーシング法をサポートし、従来のサーフェスデータでは表現が困難であった対象の可視化をすることも可能である。

This paper describes the architecture of *ReVolver*/C40 which is a high-speed volume rendering machine and prototype. *ReVolver*/C40 has a three stage parallelized macro-pipeline structure and the volume memory structure which allows to access all voxels on a ray of any direction. It provides high-speed image generation with tranlucent volume by perspective projection. These images are useful to analyze scientific computation results. It also has a mechanism for ray-traced visualization of volume data in order to visualize obscure boundary objects which are hardly expressed in surface model.

1 はじめに

ボリュームレンダリングは、3 次元空間を単位立方体 (以下では、ボクセルと呼ぶ) で構成されていると考え、この空間を 2 次元のスクリーンに投影する技法である。この技法は内部情報を持つデータの可視化が行えることから、従来、医療分野における人体データの可視化を重要な対象としていた。このため、専用並列計算機の多くは主に医療画像生成を目的とし、平行投影法による不透明ボリュームの可視化に重点が置かれて来た。

しかし近年では、構造解析や流体計算をはじめとする科学技術計算の結果の可視化による解析手法としても、ボリュームレンダリングは重要度を増してきている。そこで *ReVolver*/C40 では、科学技術計算の結果の可視化にも対応するため、遠近法による半透明ボリュームの表示、レイトレーシングによる画像生成を目的とする。*ReVolver*/C40 は、サンプリング方法を単純化し、それに合わせた 3 次元メモリ構造により、これらの目的を実現している。

本稿では、2節で *ReVolver*/C40 の概要について述べた後、3、4、5節で *ReVolver*/C40 を構成する各ステージの機能、構成について述べ、6節で現在試作しているプロトタイプについて述べる。

2 *ReVolver*/C40 の概要

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. 不透明/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

これらの目標達成のために、2.1節、2.2節で述べるような特徴を有している。

2.1 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、視線に沿って等間隔にボクセルをサンプリングしていた (図 1(a) 参照)。このため、ボリューム空間を分割して複数のメモリバンクに格納すると、同一視線上の複数のボクセルが同一メモリバンクにある場合があり、高速化の妨げとなってきた。この問題に対して、従来の専用並列計算機では、視線の方向の制限を加える等により対処していた [3]。

これに対して、*ReVolver*/C40 では、視線ベクトルの要素の絶対値のうち、最大値を持つ座標軸を主軸と呼び、この主軸の方向に等間隔なサンプリングを行う主軸等間隔サンプリング法を採用した (図 1(b) 参照)。この主軸等間隔サンプリング法の長所、短所を以下にまとめる。

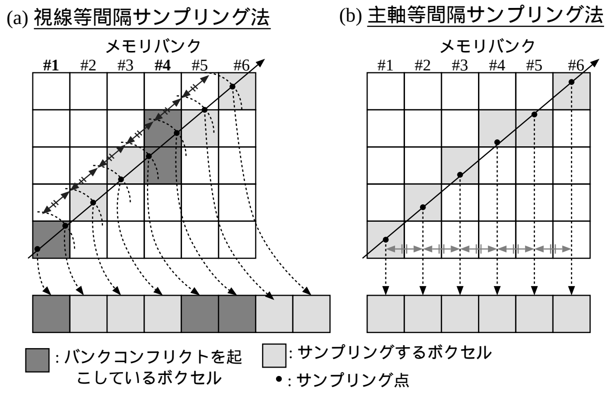

図 1: サンプリング方法の違い

長所: 主軸と垂直な平面毎に 1 点ずつサンプリングするため、ボリューム空間を主軸に垂直な平面毎にボリューム空間を分割して、別々のメモリバンクに格納すれば、バンクコンフリクトが生じない [1]。

The Parallel Computer for Volume Rendering
—— *ReVolver*/C40 ——
Yuji Tsushima,Akinori Nakayama,Tomotaka Ogino,Jin Xidu, Shin-itchiro Mori,Hiroshi Nakashima, and Shinji Tomita
Faculty of Engineering, Kyoto University
‡ current affiliation: Hitachi,Ltd.

長所: 半透明ボリュームを扱う際、読みだしたボクセルを並べ直す必要が無い。

長所: 連続モデルでの tri-linear 補間を行うに際して、従来はボクセル値が 8 つ必要だったのが 4 つで済み、演算量を減らすことができる。詳しくは 4.1節で述べる。

短所: サンプリング間隔が通常と異なるため補正が必要となる。詳しくは 4.1節で述べる。

2.2 3次元ボリュームメモリの構造

任意方向の視線では x, y, z軸のいずれもが主軸になり得る。このため、主軸等間隔サンプリング法を用いて任意方向の視線に対してバンクコンフリクトを無くすためには、図 2のように、ボリューム空間を x, y, z軸に垂直な平面群に分割し、この各平面群に対して $1, 2, \cdots, n$ と番号を付け、同一番号を有する平面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、座標 (p, q, r) のボクセルがメモリバンク p の X、qの Y、および rの Z領域に 3 重化されて格納されることになる。

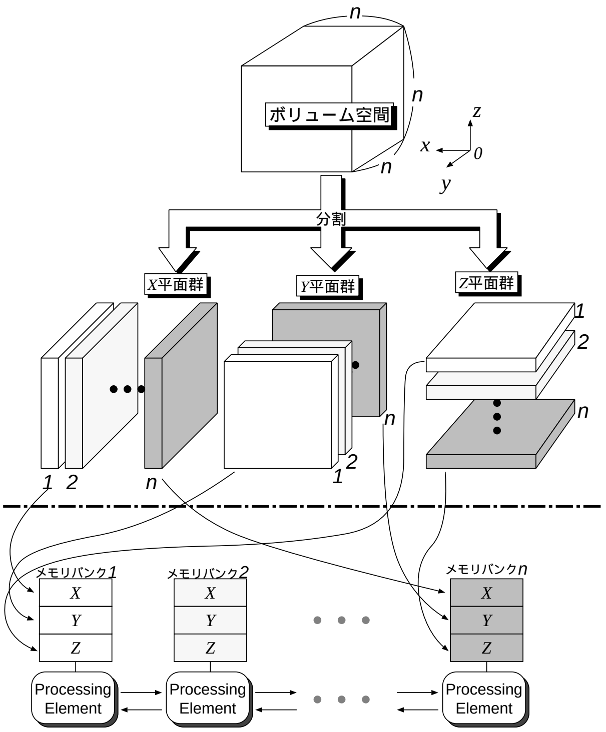

図 2: ボリューム空間の分割と格納方法

この構成では、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、任意方向の視線に対してバンクコンフリクトなしで視線上の全ボクセルを読み出せることを保証できる。これにより、半透明ボリューム表示 (目標 2) のように視線上の全ボクセルが必要となる処理に対応できる。また、遠近法 (目標 3) やレイトレーシング法 (目標 4) による画像生成のようにピクセル毎に視線の方向が異なるような処理にも対応可能となる。

さらに、高速表示 (目標 1) の達成のために *ReVolver*/C40 では、個々のメモリバンクに Processing Element(PE) を配置し、ピクセル値の計算を並列に行い高速化を図っている。

2.3 全体構成

ReVolver/C40 は以下の 4 つの要素から構成されている。

1.SCU($\dot{S}ystem\ \dot{C}ontrol\ \dot{U}nit$)

システム全体の制御を行い、ホストマシンからの指示に従って、他の構成要素に対してコマンド/データを転送する。

2.視線生成ステージ($\dot{R}ay\ \dot{C}asting\ \dot{S}tage$)

ボリュームレンダリング時 (以下では、単に通常時と呼ぶ) では、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 (以下では、単に視線と呼ぶ) を生成し、ピクセル値計算ステージへ送る。また、レイトレーシングを行う時は通常時の処理に加えて、シェーディングステージから与えられる反射/屈折光 (以下では高次光と呼ぶ) も視線としてピクセル値計算ステージへ送出する。

詳しい構成等については、3節で述べる。

3.ピクセル値計算ステージ($\dot{P}ixel\ \dot{C}alculation\ \dot{S}tage$)

ボリュームデータと色・透明度テーブルを保持しており、通常時には視線生成ステージからの視線情報に基づいてボクセルをサンプリングし、このボクセル値からピクセル値を計算するとともに、シェーディングに必要な depth も求める。また、レイトレーシング時は、ピクセル値計算に加えて視線とオブジェクト表面の交差判定、交点での法線計算の必要なデータ生成を行う。

詳しい構成等については、4節で述べる。

4.シェーディングステージ($\dot{S}hading\ \dot{S}tage$)

通常時は、depth gradient shading をはじめとしたフィルタ処理を行う。また、レイトレーシング時は、フィルタ処理は行わず視線とオブジェクト表面の交点での法線計算を行い、高次光の方向の計算を行う。

詳しい構成等については、5節で述べる。

これらの各ステージは図 3に示すように接続されており、また、各ステージの主な構成部品として汎用の DSP を使用している。このため、非常に汎用性が高くなっていることも *ReVolver*/C40 の特徴の 1 つである。

以下、本稿ではボリューム空間は 512^3voxel、スクリーン上のピクセルは 512^2pixel であり、ボクセル値は 1Byte で表されているものとする。

3 視線生成ステージ

3.1 通常時の機能・処理

視線生成ステージは、以下の SCU からのデータを基にして視点からスクリーン上のピクセルを通る視線のデータを作成するステージである。

- **視点:** 視点の位置ベクトル ($\vec{E}$) を表す。
- **スクリーン:** 以下の 3 ベクトルにより表される。

$\vec{S_b}$ $\cdots$ 視点からスクリーンの左上へのベクトル
$\vec{S_v}$ $\cdots$ スクリーンの水平方向の単位ベクトル
$\vec{S_h}$ $\cdots$ スクリーンの垂直方向の単位ベクトル

ここで、スクリーン座標が (sx, sy) のピクセルを通る視線$\vec{R}$と上記のデータの関係を図 4に示す。また、図 4に示される記号を用いて、視線生成ステージが作成する視線データの定義、機能とこれらの作成方法を表 1に示す。ただし、表 1の作成方法では主軸を x 軸としており、ベクトル$\vec{A}$の要素を $A.x, A.y, A.z$と表している。

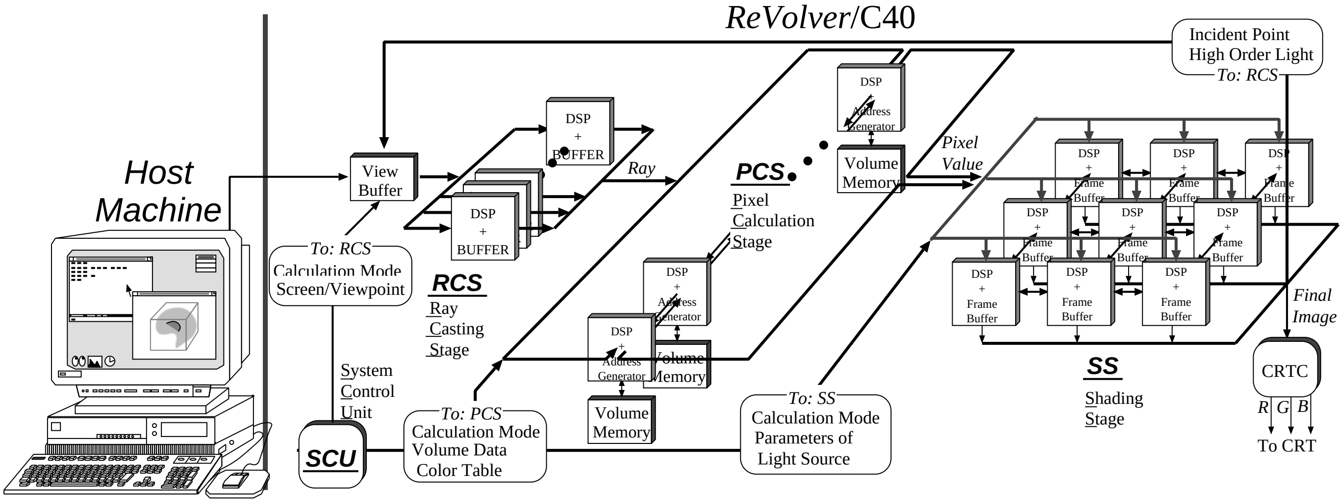

図 3: *ReVolver*/C40 の全体構成

データ名 (記号)	定義 ·機能	作成方法	サイズ
基本ベクトル (**BV1,BV2**)	視線の主軸成分を 1 とした時の残りの成分を表す。サンプリングするボクセルのアドレス計算に使用。	$\vec{R} = \vec{E} + \vec{S_b} + sx \times \vec{S_h} + sy \times \vec{S_v}$ $\mathbf{BV1} = R.y/R.x;$ $\mathbf{BV2} = R.z/R.x$	9bit × 2
基本距離 (**BLEN**)	基本ベクトルの長さを表す。depth gradient shading で使用。	$\mathbf{BLEN} = \sqrt{BV1^2 + BV^2 + 1}$	32bit
主軸名 (**MN**)	生成した視線ベクトルの要素の内最大値を持つ座標軸を表し、サンプルするボクセルのアドレス決定に使用する。	$\mathbf{MN} = \begin{cases} 01(\text{主軸} = x) \\ 10(\text{主軸} = y) \\ 00(\text{主軸} = z) \end{cases}$	2bit
主軸方向 (**MD**)	視線ベクトルの主軸成分の符号の正負により 0/1 をとる。これは、ピクセル値の計算順序を示す。	$\mathbf{MD} = (R.x > 0)?\ 0 : 1$	1bit
マスク開始位置 (**MASK**)	視点がボリュームの内部にある場合やボリュームの断面の表示において、ピクセルに反映するボクセルの選択に使用する。	if $0 < E.x < 511$ then $\mathbf{Mask} = E.x$ else $\mathbf{Mask} = (\mathrm{MD} > 0)?\ 511 : 0$	9bit
入射点 (**IP1,IP2**)	視点を延長した時、視線の主軸成分がボリューム空間の幅 (0 ~ 511) に入った時点の主軸以外の座標成分を表す。サンプリングするボクセルのアドレス計算に使用される。	if 511>E.x>0 then $\begin{cases} \mathbf{IP1} = E.y; \\ \mathbf{IP2} = E.z; \end{cases}$ else $\begin{cases} \mathbf{IP1} = E.y - E.x \times BV1; \\ \mathbf{IP2} = E.z - E.x \times BV2; \end{cases}$	9bit × 2
入射距離 (**ILEN**)	視点と入射点の距離を表す。depth gradient shading で使用。	$\mathbf{ILEN} = \sqrt{\begin{array}{l}(MD * 511 - E.x)^2 + \\ (IP1 - E.y)^2 + (IP2 - E.z)^2\end{array}}$	32bit
スクリーン座標 (**PXL.x,PXL.y**)	視線の通るピクセルのスクリーン上の座標を表す。シェーディングステージがピクセル値を取り込むのに使用する。	$\mathbf{PXL.x} = sx$ $\mathbf{PXL.y} = sy$	9bit × 2

表 1: 視線生成ステージの作成するデータ

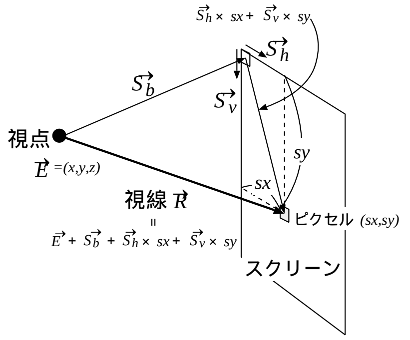

図 4: 視線生成ステージの受け取るデータ

3.2 レイトレーシング時の機能 ·処理

レイトレーシング時、視線生成ステージは通常時の処理に加えて高次光のデータも作成し、これを視線データとしてピクセル値計算ステージへ送出する。このため、高次光の方向を計算するシェーディングステージから以下のようなデータを受け取る。

- **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の表面と交差する点。高次光の始点となる。
- **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。

高次光のためのデータを作成するには、高次光の方向と交点をそれぞれ$\vec{R}$、$\vec{E}$として、表 1で示した通常時の視線作成の処理を行えばよい。

3.3 視線生成ステージの構成

3.1節、3.2節で示した処理を行う視線生成ステージは以下の要素から構成されており、図 5のように接続されている。

- **PE**: SCU/シェーディングステージからデータを受け取り視線/高次光のデータを作成する DSP(TMS 320C40)。

- **View Buffer:** ピクセル値計算ステージの全 PE で共有され、SCU/シェーディングステージからのデータを保持する 256KB のバッファ。
- **Ray Buffer:** ピクセル値計算ステージへの視線を送出するための 128KB のバッファ。

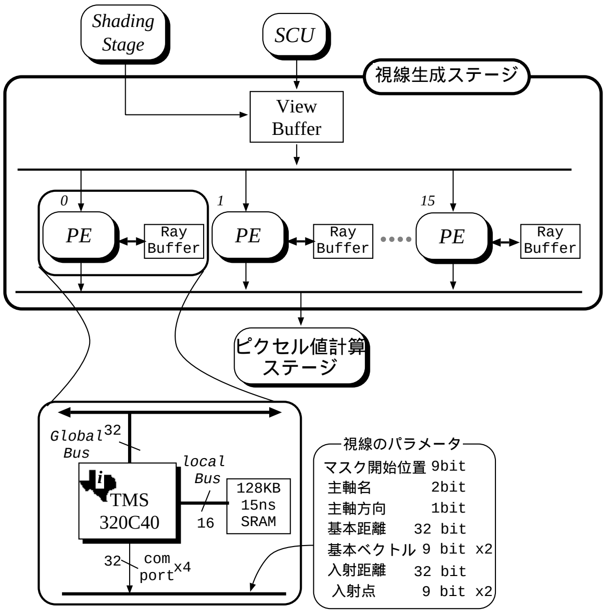

図 5: 視線生成ステージの構成

視線生成ステージの各 PE では通常時の処理では、1 スキャンライン分の視線生成を処理単位としている。このため、未処理スキャンラインの番号を View Buffer に置き、全 PE で共有する。各 PE では未処理スキャンラインの番号を獲得した場合、未処理スキャンラインを更新し、該当するスキャンライン上のピクセルを通る視線データを作成する。作成したデータはピクセル値計算ステージへ送出されるまでの間、Ray Buffer で蓄えられる。この視線データは表 1に示すように、24Byte で格納できるため、Ray Buffer には約 5K ピクセル分の視線を保持できる。

また、レイトレーシング時は高次光のデータ作成も行うので、図 5のようにシェーディングステージを接続し、View Buffer を介して、高次光データ授受を行う。ここでピクセル値計算ステージでは、通常のスキャンラインより高次光のデータ作成を優先的に処理しなければならない。これは、高次光の数が急激に増加する可能性があるためである。

表 1で示した視線データの作成に必要な DSP の命令数は、1 視線あたり 67 命令である [7]。今回使用する DSP のサイクル時間が $20ns$ であることから、仮にリアルタイム (1/30 秒) で 512^2本の視線を生成するとしても、約 10 個の DSP を並列動作させればよいことになる。したがって、このステージには 16 個の PE を配置すれば充分である。

なお、View Buffer へのアクセスは高々 1.2MB/sec 程度であり、バス競合により PE の並列動作が妨げられる心配はない。

4 ピクセル値計算ステージ

4.1 通常時の機能・処理

通常時のピクセル値計算ステージは、視線生成ステージからの視線データをもとにして、その視線上にあるボクセルをメモリから読み出し、このボクセル値を色・透明度に変換してからピクセル値の計算を行う。

ピクセル値計算ステージが作成する以下のデータは、シェーディングステージへ送出され、シェーディング処理を施されてから CRT に出力される。

- ピクセル値

  ReVolver/C40 では、半透明ボリュームの表示を行うため、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、ピクセル値は式 (1) により計算される [2]。

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

  ただし、$c(v_i), \alpha(v_i)$ はそれぞれボクセル値 v_iに対する色、透明度を表している。

- **depth**

  視線が最初に交差する可視ボクセルの座標の主軸成分。シェーディングステージでピクセルの傾斜の計算に使用される。

ReVolver/C40 では、ピクセル値計算時のボクセル値から色・透明度の変換に関して、あらかじめ色・透明度のテーブルを用意しており、読み出したボクセル値を用いてこのテーブルを参照することにより、式 (1) に必要な $c(v_i), \alpha(v_i)$ を求めている。ボクセル値は 1Byte で表されているので、この色・透明度テーブルは 256 エントリから構成されている。また、テーブル参照の方法として以下の 2 通りを用意している。

- 直接: ボクセル値を直接インデックスとして、色・透明度テーブルを参照する。 通常はこちらを使用する。
- 差分: 隣接ボクセル値の差分をインデックスとして、色・透明度テーブルを参照する。視線上でボクセル値の変化が急な箇所を強調するのに用いる。

また *ReVolver*/C40 では、より高精細な画像を生成するために、ピクセル値計算時のオプションとして、連続モデルのボクセルによる計算を用意している。連続モデルのボクセルでは、ボリュームを構成する単位立方体の頂点にボクセル値が存在し、ボクセル内部の点のボクセル値は頂点の値から tri-linear 補間により求められる [2]。しかし、*ReVolver*/C40 では、主軸等間隔サンプリング法により、サンプリング点は必ずボクセル側面に位置するため、補間にはこの側面上の 4 頂点の値のみでよいことになる。したがって、演算量、ボクセル値の参照回数が半分になり高速実行が可能となる。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 で行う補間の式を式 (2) に示す。ただし、式 (2) で用いる記号は図 6で示すものである。

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

また、主軸等間隔サンプリング法には、視線が軸と平行でない時サンプリング点の間隔が従来の方法に比べて大きくなるという性質がある。そこで、*ReVolver*/C40 ではピクセル値計算時にサンプリング間隔の補正を行うオプショ

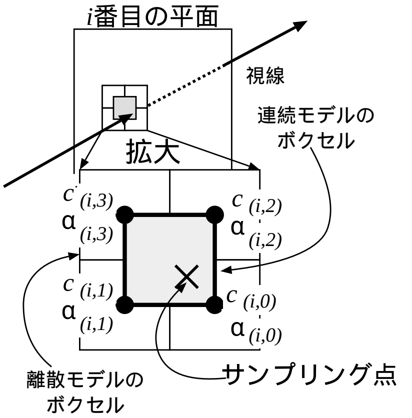

図 6: 連続モデルのボクセルの補間

ンも用意する。$ReVolver$/C40 で行う補正は、補正前の色・透明度を c, α とし、補正後の色・透明度を c', α' とすると、以下の式 (3) のように表される。

$$\left.\begin{array}{rcl} c' & = & c \cdot \alpha^{\Delta D - 1} \\ \alpha' & = & \alpha^{\Delta D} \end{array}\right\} \quad (3)$$

しかし、式 (3) を直接評価するのは PE の負担が大きいため、以下のような近似式を用いることで対処している。

$$\alpha^{\Delta D} = 1 + \Delta D \cdot (\alpha - 1) + \frac{\Delta D \cdot (\Delta D - 1)}{2} \cdot (\alpha - 1)^2 \quad (4)$$

4.2 レイトレーシング時の機能・処理

レイトレーシング時は通常のピクセル値計算に加えて、ボリューム内のオブジェクトの表面と視線の交差判定も行う。また、ピクセル値計算について通常時と異なるのは、視線と表面が交差している場合、交点の直前のボクセルで式 (1) の計算を打ち切ることである。

以下では、ボリュームデータを用いてレイトレーシングを行う際に問題となる点を 2 つ挙げ、$ReVolver$/C40 での対処方法について述べる。

問題点 1 —— 交差判定 ——

ボリュームデータは通常のサーフェスデータと異なり、オブジェクトの境界面の決定が困難なため、交差判定も困難となる。また通常時のように視線が最初に交差する無色・透明でないボクセルを交点とすると、高次光が連続して反射屈折を繰り返すことになるので、使用できない。

そこで、$ReVolver$/C40 では視線上の隣接するボクセルの値の差をとり、この差が一定値以上であれば境界面の候補とし、これらの候補の中から視点に最も近いものを表面として選択するようにしている [1]。

問題点 2 —— 法線計算 ——

レイトレーシングでは、高次光の方向計算のために交点での法線計算は必要不可欠である。従来のボリュームデータを使ったレイトレーシングでは交点の近傍のボクセル値から法線を計算していた [6]。しかし、$ReVolver$/C40 ではボリュームデータを複数のメモリバンクに分割して格納し、バンク毎に PE を配置しているため、この方法では交点の近傍を含むメモリバンクを担当している PE に処理が集中してしまい、その PE が全体を律速してしまう危険性がある。

そこで、この問題を回避するために $ReVolver$/C40 では、視点を主軸以外の軸方向に移動させ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て、各平面で平行視線上のボクセルもサンプリングする (図 7参照)。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する。

平行視線に対しては、視線と同様の交差判定のみを行う。これにより、視線、平行視線を合わせて 5 つの交点が得られ、これらの交点を図 7のように A,W,E,N,S とすると、点 A での法線ベクトルは以下の式 (5) により計算できる。

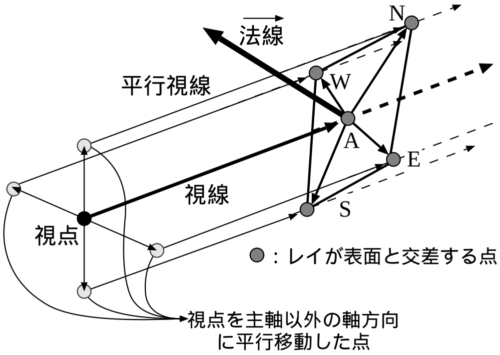

図 7: 法線計算

$$\overrightarrow{法線} = \overrightarrow{AN} \times \overrightarrow{AW} + \overrightarrow{AW} \times \overrightarrow{AS} + \overrightarrow{AS} \times \overrightarrow{AE} + \overrightarrow{AE} \times \overrightarrow{AN} \quad (5)$$

(但し、$\vec{A} \times \vec{B}$ は $\vec{A}, \vec{B}$ の外積を表す)

$ReVolver$/C40 で採用している方法は、ピクセル値計算のために参照する全ての平面で参照パターンが同じになり、交点近傍へボクセルのアクセスが集中しなくなる。このため、特定 PE への処理の集中がなくなり、律速の心配はなくなる。

4.3 ピクセル値計算ステージの構成

4.1節、4.2節の処理を行うピクセル値計算ステージは以下の要素から構成されており、図 8に示すように接続されている。このステージでは、ピクセル値計算を多数の PE によりパイプライン処理することで高速化を図っている。

- **PE**: 次の 3 要素により構成される。
  1. **DSP(TMS 320C40)** ··· ピクセル値計算、交差判定を行う。TMS 320C40 には Local Bus、Global Bus があり、それぞれに 3D Volume Memory、Buffer を接続する構成を採っている。また、隣接 PE との通信には DSP の com port を使用する [7]。
  2. **Address Generator** ··· サンプルするボクセルのアドレス計算を行う。FPGA で構成する。
  3. **Buffer** ··· PE 間通信のための 256KB のバッファ。
- **3D Volume Memory**: 2.2節の方法によるボリュームデータの格納。

これらの要素による、ピクセル値計算ステージの PE での処理の流れは次のようなものである。(i)Address Generator が前段からのデータをもとに読み出すボクセルのアドレスを生成する。(ii) アドレスを 3D Volume Memory へ入力しデータが出力される。(iii)DSP が 3D Volume Memory からの出力を取り込んで、4.1節で述べた方法により色・透明度に変換し、隣接 DSP からの色・透明度を用いて式 (1) を評価し、次 PE へ送信する。

次に、このピクセル値計算ステージの PE 間のパイプライン処理で、PE 間を流れるデータを大別すると以下の

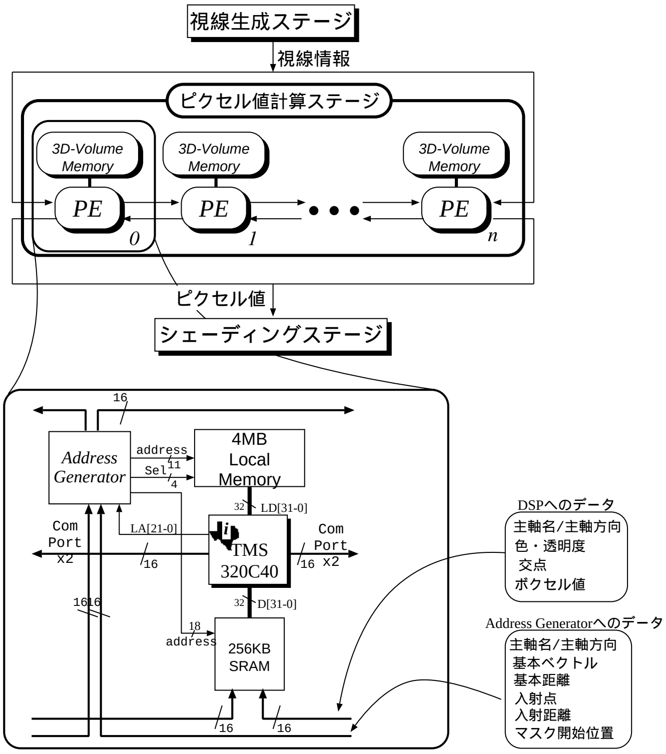

図 8: ピクセル値計算ステージの構成

(**A**) と (**B**) になる。以下では、このデータを用いで DSP と Address Generator が行う処理とテータ自身について述べる。

(A) DSP 間を流れるデータ

i 番目の DSP から $i+1$ 番目の DSP へ流れるデータをまとめると表 2のようになる。各 DSP では、自 PE で読み出したボクセル値を使い上記の (iii) の処理を行い、表 2 の各データを更新して次 PE へ渡す。

表 2: ピクセル値計算ステージの DSP 間で流れるデータ

名称	定義 ・機能 ・使用目的	サイズ
主軸名 主軸方向	ボクセルを読み出すメモリ領域とピクセル値計算の計算方向を示す。	3 bit
色	式 (1) を i まで評価した値。 $\sum_{j=0}^{i}(1-\alpha(v_j))c(v_j)\prod_{k=0}^{j-1}\alpha(v_k)$	16bit × 3
透明度	i までの透明度の積。 $\prod_{j=0}^{i}\alpha(v_k)$	16 bit
・通常時		
ボクセル値	自 PE で読み出すボクセル値。ピクセル値計算、交差判定に使用する。	8 bit
交点	視線上で交差が起きた時の PE 番号シェーディングに使用する。	9 bit
・レイトレーシング時		
ボクセル値	自 PE で読み出す視線と平行視線のボクセル値。使用目的は通常通り。	40 bit
交点	視線と平行視線上で交差が起きた時の PE 番号。法線計算に使用する。	45 bit

(B) Address Generator 間を流れるデータ

視線生成ステージで作成されたデータ (表 1) の入射点のみが以下の式 (6) のように変更されながら Address Generator 間を流れる。

$$i+1 \text{ への入射点} = i-1 \text{ から入射点} + \text{基本ベクトル} \quad (6)$$

以下に、Address Generator で行われる処理を示す。

(a) アドレス計算 ···視線データから自 PE で読み出すべきボクセルのアドレス計算を行う。ここで、生成するアドレスは表 1 の記号を使って表すと以下にようになる。

$$MN << 20 + (IP1 + BV1) \times 512 + (IP2 + BV2)$$

この $(IP1 + BV1)$,$(IP2 + BV2)$ が式 (6) の右辺に相当する。

(b) マスク調べ ···マスク開始位置と PE 番号の比較を行い、PE 番号の方が小さい場合は、不要なボクセルであるので、DSP のピクセル値計算に影響を及ぼさないようにするために、無色 ・透明のデータのアドレスを出す。

(c) Local Bus 制御 ··· DSP の Local Bus を図 9 のような構成とし Local Bus を制御することで、DSP の Byte アクセスを効率化を図る。この機構により必要なボクセル値が必ず最下位 Byte にくるため、DSP は LBU0(Load Byte Unsigned) 命令によりアクセスするだけでよくなる [7]。

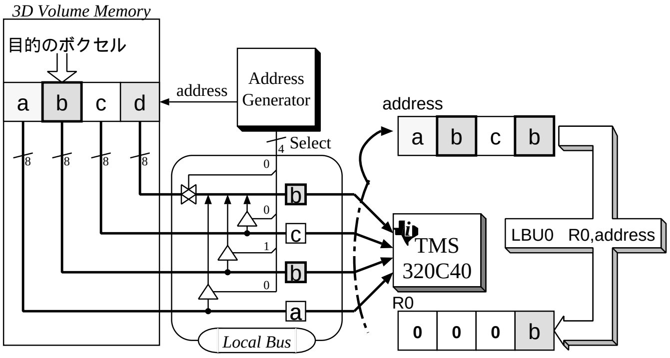

図 9: Local Bus の構成

5 シェーディングステージ

5.1 通常時の機能、処理

シェーディングステージは通常時には、ピクセル値計算ステージからのピクセル値と depth を使って depth gradient shading を行う。depth gradient shading はピクセル毎に近傍のピクセルの depth を使い、そのピクセルの傾斜を求め、光源に対しどのような向きにあるかを計算し陰影をつける処理である [5][4]。ただし、ピクセル値計算ステージからの depth は、視線が交差する最初のボクセルの座標の主軸成分を表しているため、depth gradient shading を行うには、式 (7) を評価した値を用いなければならない。

$$\text{入射距離} + depth \times \text{基本距離} \qquad (7)$$

スクリーン上の (x, y) の depth を $d_{(x,y)}$、スクリーンと視点の距離を 1 とした時のピクセル間の距離を dp で表すと、このピクセルのスクリーンの x 方向の傾斜を求める方法を以下に示す。

1. $\Delta D_k = d_{(x+k+1,y)} - d_{(x+k,y)}$を $k = 0, \pm1, \pm2, \pm3$ に対して計算する。
2. 全てのΔD_kに対して、ΔD_0を含みΔD_0と符号が異ならない最大の連続した k範囲を求める。この kの最大値を k_0、最小値を k_1とする。
3. $D = d_{(x+k_0,y)} - d_{(x+k_1,y)}$を求める。
4. $\theta_x = \arctan(\frac{D}{|k_0-k_1+1|\cdot dp\cdot d_{(x,y)}})$ を計算すれば、スクリーンの水平方向の傾斜角が求まる。

この処理をスクリーンの y方向に対して行えばピクセル (x, y) の傾斜ベクトルが計算できる。

5.2　レイトレーシング時の機能、処理

レイトレーシング時は depth gradient shading を行わずに、4.2節で示した交点での法線ベクトルの計算方法を行う。シェーディングステージでは、ピクセル値計算ステージから視線、平行視線の交点を受け取り A,W,E,N,S のボリューム内座標を計算し、これらから式 (5) の計算を行う。

高次光の方向の計算は、法線ベクトルを正規化したものを$\vec{N}'$で表すと、以下のような式 (8) により求められる。

$$\left.\begin{array}{lcl} \text{反射光} & = & 2\vec{N}' + \frac{\vec{R}'}{|\vec{R}'\cdot\vec{N}'|} \\ \text{屈折光} & = & k_f(\vec{N}' + \frac{\vec{R}'}{|\vec{R}'\cdot\vec{N}'|}) - \vec{N}' \end{array}\right\} \quad (8)$$

$$\left(\begin{array}{l} \vec{R}'\text{は視線ベクトルの単位ベクトルを表す。} \\ \text{入射側、透過側の屈折率を } n_1\text{、}n_2\text{で表すと、} \\ k_f = \left((\frac{n_1}{n_2})^2|\frac{\vec{R}'}{\vec{R}'\cdot\vec{N}'}|^2 - |\frac{\vec{R}'}{\vec{R}'\cdot\vec{N}'} + \vec{N}'|\right)^{-1} \end{array}\right)$$

5.3　シェーディングステージの構成

5.1節、5.2節の処理を行うシェーディングステージは、以下の要素から構成されており、図 10のように接続されている。

·**PE:** 以下の 3 つの要素により構成されている。PE 間結合は画像処理に適したメッシュ結合を採用している。

- **DSP(TMS 320C40)** ··· 傾斜の計算や法線、高次光の方向計算を行う。
- **SRAM** ··· フレームメモリとして使用する。ピクセル値計算ステージからの受信のためにダブルバッファリングするため 2 バンク準備する。
- **VRAM** ··· CRT へ出力するためのフレームメモリ。

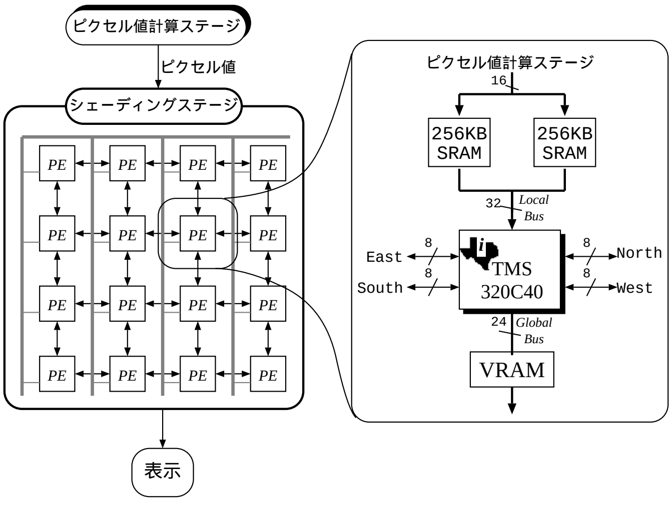

図 10: シェーディングステージの構成

通常時の depth gradient shading の処理で PE 間通信が必要のないようにするために、図 11のようにシェーディングに必要な領域を PE 間で共有して持つようにスクリーンを分割する。このシェーディング処理に必要な命令数は、1 ピクセル当たり約 140 命令である [7]。したがって 1/30 秒で 512^2ピクセルを処理するには、22 個程度の PE を並列動作させればよいことになる。よって、このステージには 32 個の PE を配置すれば充分である。

また、レイトレーシング時の処理に必要な命令数は約 300 命令必要であり、レイトレーシング時のピクセル値計算は通常時の 3 倍程度時間がかかるので (6.1節参照)、高次光計算によるパイプラインの乱れはないと考えられる。

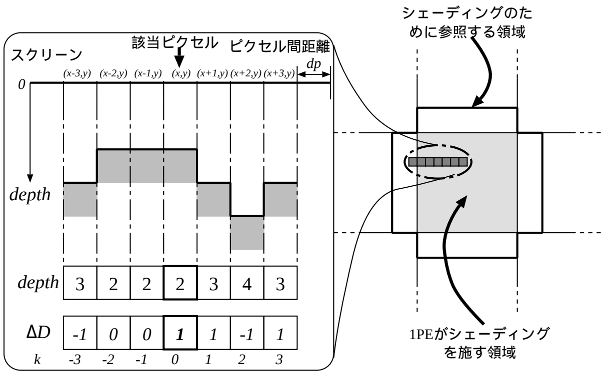

図 11: データ分割方法

6　プロトタイプの性能と機能拡張

6.1　プロトタイプでの性能

プロトタイプでは、ピクセル値計算ステージの性能に応じて他のステージの PE 数を決定するため、ピクセル値計算ステージの性能が非常に重要となる。そこで、本節ではピクセル値計算ステージでの各種の計算方法について性能予測を行い、最後に視線生成ステージとシェーディングステージに必要な PE 数を求める。ただし以下では、ボリューム空間は (ピクセル値計算ステージの PE数)^3**voxel** であり、ピクセル数は 512^2**pixel** であるとして考える。

ピクセル値計算ステージの PE での処理時間を大別すると、ボクセル値のアクセス時間、ピクセル値計算時間、データの送受信の時間に分けられる。しかし、ボクセルのアクセスとピクセル値計算等は Address Generator と DSP でパイプライン的に行うことができるので、メモリアクセス時間と DSP の処理時間について考えればよい。

そこで、離散モデルに関して通常時とレイトレーシング時について、メモリのアクセス時間と DSP での処理に必要な時間を表 3に示す。ただし、表中の () 内の数字はアクセスするボクセル数を表し、メモリアクセス時間は使用する DRAM のアクセス時間を $160ns$、Fast Page Mode でのアクセス時間を $80ns$ としている。また、DSP の処理時間は、アセンブラを用いてプログラムを記述しその命令数から時間を求めている [7]。

表 3: 各モードでの処理時間

	メモリアクセス	DSP の処理
通常時	$160ns(1)$	半透明$360ns$ 不透明$260ns$
レイトレーシング時	$640ns(5)$	$1.84\mu s$

通常時とレイトレーシング時では、メモリアクセスより DSP の処理時間の方が長いため、DSP の処理時間でピクセル値計算ステージの性能が決まっている。したがって、通常時の半透明ボリュームでの画像生成速度は、$360ns \times 512^2 =$ 約 $94.4ms/frame$ であり、不透明ボリュームでの画像生成速度は、$260ns \times 512^2 =$ 約 $68.1ms/frame$ である。

次に、レイトレーシング時では、全視線で 2 回反射屈折が起きるとして考えると、処理すべき視線が$\sum_{i=0}^{2} 2^i =$ 7 倍になるため、画像生成速度は $7 \times 1.84\mu s \times 512^2 =$ 約 $3.38sec/frame$ となる。

また、高精細な画像を生成するための連続モデルによる画像生成では、DSP で式 (2) の補間計算を行うため、サンプリング点の座標を DSP で知る必要がある。このため、メモリアクセスとピクセル値計算をパイプライン化できず全て DSP で行わなければならない。そこで、連続モデルに基

づく画像生成のプログラムを作成し試算したところ、1 ピクセルを計算するのに約 $1600ns$ 必要となることが分かった [7]。したがって、$1600ns \times 512^2 =$ 約 $0.42sec/frame$ で画像生成できることになる。

ReVolver/C40 の性能を、計算モードと扱うボクセルのモデルについてまとめると以下の表 4のようになる。

表 4: *ReVolver*/C40 の性能　(単位:per frame)

計算モード	離散モデル	連続モデル
通常時	68.1ms	0.42sec
レイトレーシング時	3.38sec	—

ここで、視線生成ステージと、シェーディングステージの PE 数について、3節、5節では 1/30 秒で 512^2pixel の処理をするのに必要な PE 数を求めていたが、プロトタイプでのピクセル値計算ステージの性能 ($68.1ms/frame$) に合わせて考えると、それぞれ 8PE、16PE でよいこととなる。

6.2　プロトタイプの規模と機能拡張

現在開発中の *ReVolver*/C40 のプロトタイプのハードウェア規模と扱うデータの大きさを表 5に示す。また、プロトタイプとしての機能拡張として、ピクセル値計算ステージの 1PE が担当する平面を図 12のように変更できるようにしている。これにより、表 5のように小規模な構成でも 512^3voxel という大きな空間を扱うことができるようになり、柔軟な処理が可能となる。

表 5: *ReVolver*/C40 の規模

ステージ名	PE 数	
	最終構成	プロトタイプ
視線生成ステージ	8	8
ピクセル値計算ステージ	512	128
シェーディングステージ	16	16

項目	扱うデータ	
	最終構成	プロトタイプ
Volume 空間	$812^{3\dagger}$	512^3
スクリーン	512^2	512^2

$\dagger : 812 \simeq \sqrt[3]{512 \times 1024^2}$

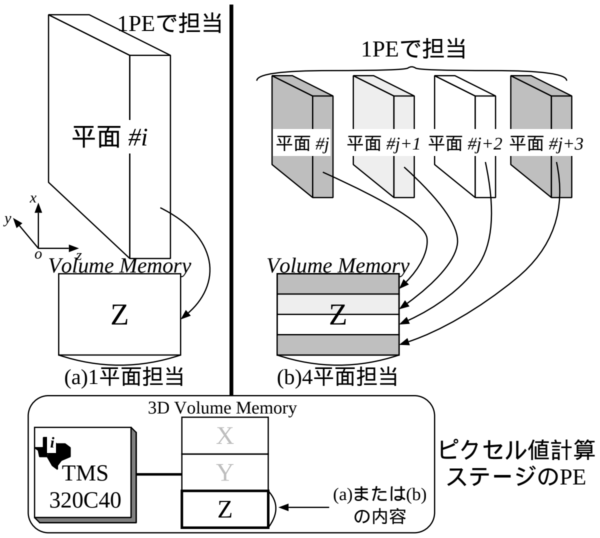

図 12: 1PE による複数平面の担当

プロトタイプでは、この機能を実現するために、ピクセル値計算時の Address Generator によるアドレス生成と DSP によるピクセル値計算を、それぞれ複数のボクセルに対して行うようにする。また、Address Generator で行われるマスク開始位置の比較は単に PE 番号と比べるのではなく、PE 番号から PE 番号+ (平面の数) までの値と比較するように構成することで実現している。

ただし、1PE で複数の平面を担当する場合は、DSP の処理が担当する平面の数に比例して増加するため、その分時間がかかることになる。また、1PE で担当できる平面の数の上限は、ピクセル値計算ステージの PE に配置する Volume Memory の大きさにより決定される。プロトタイプでは、512^3voxel のレンダリングを 128PE で行うことを考えて、各 PE に 4MB を配置している。

7　まとめ

本稿では、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これを構成する各ステージの処理と構成ならびに現在開発中のプロトタイプについて述べてきた。

ReVolver/C40 では、ボクセルのサンプリング方法を単純化し、これに合わせたボリュームメモリ構成を採用し、3 ステージマクロパイプラインと各ステージの並列処理によって高速化している。これにより、半透明ボリュームでの性能は、$94.4ms/frame$ であり、不透明ボリュームの表示では $68.1ms/frame$ で画像生成を行うことができる。また、各ステージの主要部品として汎用の DSP を使用していることから、様々な応用にも対応できる構成となっている。

今後の課題として、プロトタイプの作成を行い、実機上での性能や画像の評価を行っていく予定である。

謝辞

日頃御討論頂く京都大学工学部情報工学教室富田研究室の諸氏に感謝致します。本研究の一部は文部省科学研究費 (一般研究 (A) 課題番号 06402057 「3 次元メモリによる実時間可視化機構を内蔵した柔構造スーパコンピュータの構成方式」) による。

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のアーキテクチャ,JSPP'94, pp89-96, 1994.

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

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

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

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

[6] 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.

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