計算機アーキテクチャ 114－4
（1995. 10. 19）

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 アーキテクチャー

金 喜都　對馬雄次†　中山明則　荻野友隆　森眞一郎　中島浩　富田眞治

京都大学工学部 (†:現在日立制作所)

本稿では，512^3ボクセル集合に対する半透明ボリューム表示、遠近法による投影、高速な画像生成を目的とした、レイキャスティングを超高速に行える専用並列計算機のアーキテクチャを提案する。それは表示図形歪みを抑える妥当な視角で3次元ボリューム空間を直接視覚化し、ユーザーに正しく認識、理解、洞察できる画像を提示する。

本計算機では、ボリューム空間を、メモリバンク群に対応するスライス群に写像することにより、512^2ピクセルをもつスクリーンの一行分 (または一列分) のピクセルを通る512本視線に交差する全ボクセルを獲得する。同時に、512個プロセッサを1次元に接続し、並列に動作させ、80ns の固定パイプラインサイクルΔTで各視線の前進ステップの計算を各プロセッサの上に同期的に完成することで、512ΔTごとに512本視線のレイキャスティングを行える。従って、1秒間に48枚動画が形成できる。

The Super-high Speed Special-Purpose Computer Architecture Based on the Pixel Parallel Processing for Volume Rendering

Xidu Jin　Yuji Tsushima†　Akinori Nakayama　Tomotaka Ogino
Shin-i chiro Mori　Hiroshi Nakashima　Shinji Tomita

Faculty of Engineering ,Kyoto University (†: Hitachi, Ltd)

This paper presents a super-high speed ray casting system for volume rendering of 512^3 volumetric datasets. In order to realizes real-time image generation from a translucent volume by the parallel or perspective projection, this system restricts the view angle within a certain range. Since the range is determined so that no distortion will be appeared on the generated image, this system can provide the users with proper understanding about the unknown 3D objects.

This system adopts a conflict free memory of 512-banks and the highly parallel one-dimensional pipelined architecture. Provided with the constant pipeline cycle of 80ns, this system processes 512 rays every 512 cycles, resulting 48 frames generation per second.

1 はじめに

従来の医療分野における立体画像の生成方法や今日の膨大な格子点に配布した科学技術計算結果の可視化による解析手法として、ボリュームレンダリングは注目を集めている。サーフェスレンダリングと異なって、ボリュームレンダリングの目的は、通常は見ることの出来ないボリュームオブジェクトの内部を探索し、従来は解析が困難であった複雑な構造や振り舞いを理解するための視覚的な洞察手法を提供することにある。

ところが、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。特に科学技術計算結果の可視化には、遠近法による画像生成と半透明なボリューム表示に必要な高速化と、長時間にわたる時系列の計算結果の動画化が更に困難である。

そこで、本研究では512^3ボクセル集合に対する半透明なボリューム表示、遠近法による投影、視点移動に伴う高速な画像生成、時間軸に沿う連続的な現象表示のための動画作成を目的とした、ピクセル並列レイキャスティングが超高速に行える専用並列計算機のアーキテクチャを提案する。

2 ボリューム映像の並列化

2.1 基本概念

以下本稿で用いる用語について説明する。

2.1.1 ボリューム・レンダリング

ボリューム・レンダリングとは、ボクセルと呼ばれる単位立方体集合の格子構造で構成されている3次元の空間内の内部情報を伴う3次元オブジェクトを、2次元のスクリーン空間に投影することにより、3次元空間を可視化する手法である。この3次元空間をボリューム空間と呼ぶ。

2.1.2 ボリューム空間とスクリーン空間

3次元格子状のボリューム空間がn^3ボクセルを持つものであるとする。それを3次元直交座標系O_{-LAB}上のL, A, B軸の$[0, n)$半開区間上に定義し、各ボクセルの8つの頂点を3次元座標系の整数座標点上に置く。ボリューム・データの処理方法に応じ、以下の2種類のモデルが定義される。

I. 離散モデル：各ボクセル値を中心とした単位立方体をボクセルと呼び、ボクセル内の任意の点データの属性値は、ボクセル値と等しく見なす。

II. 連続モデル：8つの頂点各々が、独自のボクセル値を持つ単位立方体をボクセルと呼び、ボクセル内での属性値は通常、3重(tri-linear)補間手法を用いて求められる。

また、対応したスクリーン空間は、n^2ピクセルを持つものとする。

2.2 ボリュームレンダリングを施す基本手法

2.2.1 レイ・キャスティング法

レイキャスティング(ray casting)法はボリューム空間の中身を覗き、半透明なボリューム表示を行う手法である。それは、視点からスクリーンの各ピクセルに対して視線を発生し、その視線に沿って不透明度が1になるまで光を吸収するボリューム内を前進する。そして、最終的にはそれまでの累積カラーを求めてそのピクセルの色とする。

2.2.2 主軸等間隔サンプリング

3次元格子状の離散ボリューム空間でレイ・キャスティングを行うため、視線上のボクセル値のサンプリングは主軸等間隔サンプリングによって行うことにする。アルゴリズムとしては、視線のサンプリングを、当該視線の方向ベクトルの成分の絶対値が最大となる座標軸(以後主軸と呼ぶ)上で、等間隔(ボクセル間の距離)で行なうようにする。即ち、あるピクセルを通る視線$Ray_i\,(i = 1 \sim n^2)$上のボクセル値のサンプリング間隔ΔD_iは次の式(1)によって決る。

$$\Delta D_i = \sqrt{\Delta L_i{}^2 + \Delta A_i{}^2 + \Delta B_i{}^2} \qquad (1)$$

ただし, $0 \le |\Delta L_i|, |\Delta A_i|, |\Delta B_i| \le 1$ 且つ
$\exists \Delta V \in \{|\Delta L_i|, |\Delta A_i|, |\Delta B_i|\}, \Delta V = 1$
ここで, ΔV は主軸サンプリング間隔でもあり、ボクセル(Voxel)の稜線の長さでもある。
故に, $1 \le \Delta D_i \le \sqrt{3}$

従って、各視線上にサンプリングされたボクセルの数は多くてもn個である。ところが, 主軸等間隔サンプリング法は、軸と平行でない視線が通常の視線等間隔サンプリング法(視線の方向に沿ってボクセルを等間隔にサンプリングする)に比べると、そのサンプリング間隔は最大で$\sqrt{3}$倍となる。このため、ピクセル値の補正が必要となる。

一方、視線等間隔サンプリングは、どの座標軸にボクセル間隔のサンプリングが出来ないことがある。ことに反して、主軸等間隔サンプリングは、少なくてもある一つの座標軸上にボクセル間隔によってサンプリングが行える。

故に、主軸等間隔サンプリングは、ボリューム空間をメモリバンク群に対応するスライス群に写像することで、ボクセル並列アクセスによるバンクコンフリクトを無くすメモリ構成が達成する基礎である。

2.2.3 ピクセル値の算出方式

主軸等間隔サンプリングに基づくレイキャスティング法によるピクセル値Pの計算式は、視線が通るボクセルv_iを、視点から近い順に$i = 0, 1, \ldots, n-1$とするとき、以下のように表現される。

$$P = \sum_{i=0}^{n-1} K(v_i) \prod_{j=0}^{i-1} \alpha(v_j) \qquad (2)$$

ただし, $c(v_i)$は、ボクセル値v_iに対応する色
$\alpha(v_i)$は、ボクセル値v_iに対応する透明度
$K(v_i) = c(v_i)(1 - \alpha(v_i))$
ただし、$K(v_i) = c(v_i)$場合もある

2.3 並列処理手法と画面生成率

2.3.1 ボリューム・レンダリングの並列処理手法

レイ・キャスティング法を用いて画像生成を行うとすれば、高速処理のための並列化手法は以下の2種類に分ける。

ボクセル並列処理 ボリューム空間で任意一つの視線と交差する全てのボクセルのデータを同時に取り出して、視線のピクセル値を計算する。

ピクセル並列処理 複数の視線を同時に追跡して各視線に交差するボクセルデータを探索しながらピクセル値を計算する。

2.3.2 メモリ構成による画面生成率

3次元ボリューム空間の n^3 個ボクセルのデータを格納してあるメモリ構成及び n^2 ピクセルからなるスクリーンに対して、仮にメモリ・サイクルΔT でメモリ構成から任意の n 個ボクセルのデータを同時に読み出されるとすると、離散モデルでボリュームレンダリングする場合には単位時間で生成できる画面の数、即ち、画面生成率は $(n^2\Delta T)^{-1}$ である。

これはボクセル並列処理にもピクセル並列処理にも正しいのである。

2.4 Pixel並列処理によるレイキャスティング

本稿で提案する計算機の最大の特徴は、ピクセル並列処理による画像生成の高速化を図ることである。ピクセル並列処理の実行は以下の説明によって行うようにする。

2.4.1 漸化式表示のレイ・キャスティング法

2.2.3節のレイ・キャスティング算式(2)より

$$\begin{cases} \alpha_m &= \prod_{i=0}^{m} \alpha(v_i) \\ K_m &= \sum_{i=o}^{m} K(v_i) \times \alpha_{m-1} \end{cases}$$

とおくと、式(2)は

$$\alpha_{-1} = 1, K_{-1} = 0 \text{ を初期値として}$$

$$\begin{cases} \alpha_j &= \alpha_{j-1} \times \alpha(v_j) \\ K_j &= K_{i-1} + K(v_j) \times \alpha_{j-1} \end{cases} \quad (3)$$

$$(j = 0 \sim n-1)$$

と漸化式で書き下せる。また、最終結果の K_{n-1} はピクセル値 P である。

従って、視線に対するピクセル値は、その視線の通るすべてのボクセルについて視点側から順にボクセルごとに式(3)を繰り返し行うことで算出される。

2.4.2 漸化式表示に適する視線追跡サンプリング

視線追跡サンプリングとは、2.2.2節の主軸等間隔サンプリングによる$\Delta D = (\Delta L, \Delta A, \Delta B)$ を視線の前進ステップとして、ボリューム空間への入射点の座標 (L_{in}, A_{in}, B_{in}) から視線方向に沿って1回に前進ステップずつ進め、座標点 $(L_{in} + j \times \Delta L, A_{in} + j \times \Delta A, B_{in} + j \times \Delta B)$ のボクセル値 v_j を $j = 0, 1, 2, \ldots, n-1$ の順にサンプリングを行うことである。

2.4.3 ピクセル並列処理によるレイキャスティング

本計算機のピクセル並列処理手法は、スクリーンの1行分(または1列分)ピクセルを通る n 本視線を視線群として、この視線群の各視線に対して、同時に視線追跡サンプリングに対応する 漸化式表示のレイキャスティングを行うようにする。具体な方法は以下のようにする。

視線群の各視線 $Ray_m (m = 1 \sim n)$ は、各々ボリューム空間への入射点から各視線に沿って、

(1) 各自の前進ステップ$\Delta D_m (m = 1 \sim n)$ に従い、一つの前進ステップに進める。

(2) これらの前進ステップに交差する各ボクセルの値 $v_{m_j} (m = 1 \sim n)$ を同時にサンプリングする。

(3) これらの $v_{m_j} (m = 1 \sim n)$ を使って属性テーブルを同時に参照することで $K(v_{m_j}), \alpha(v_{m_j}) (m = 1 \sim n)$ を求める。

(4) $K(v_{m_j}), \alpha(v_{m_j}) (m = 1 \sim n)$ を各視線による漸化式(3)に代入し並列に計算する。

以上の(1)～(4)を、$j = 0$ から視線がスライス群を出るまでまたは不透明度が1になるまで繰り返して計算していく。最後に求めた $K_m (m = 1 \sim n)$ は各視線のピクセル値とする。

また、以上の(1)～(4)を前進ステップの計算を呼ぶ。

3 ピクセル並列処理の実施

3.1 視角制限のピクセル並列処理とその妥当性

視点は立方体ボリューム空間の 90^0頂点を連結する対角線の延長線にある場合、本計算機のピクセル並列処理で生成された画像の視角は本計算機のメモリ構成によって最小 54^0準広角[‡]に制限される。

一方、オリジナルのデータが変更されたり誇張された表現のための映像と異なって、科学技術計算結果の視覚化は主として専門家自身が解析の「過程」で正しい判断を得て複雑現象を正確につかめ、正確に理解できるためである。よって、対象の直接見えない内部構造を歪み無しで客観的に反映できる映像を追求し、可視化を客観的な視覚効果に合致させることは大事なことである。

広角的な視角による平面スクリーンへの映像は2つの欠点がある。

[‡]準広角：カメラ撮影により EF35mm のレンズ即ち 54^0視角に対応する。$\tan(\frac{54^0}{2}) = 0.51$ なので, 視点からスクリーンまでの距離がスクリーンの辺の長さと殆ど等しい。

(1) 広角的な視角で,平面スクリーンの各ピクセルに制約されるレイキャスティングをすると,ボリューム中部空間を通過する視線の数が広角ほど少なくなる。返って、ボリュームの周辺空間を通る視線の数が広角ほど多くなる。従って、ボリューム中身の情報欠落に導くようになっている。

(2) 広角的な視角に導かれた歪みの映像は,人間が対象の直接見えない内部構造をはっきり見分ける(認識する)ことを妨げる。そこで、可視化の目的に背くことになる。

よって,歪んでしまうことが起こらずボリューム中身の情報欠落を防止することで画面の中心部も周辺部も均一で上質の描写が得られるように視角を制限しなければならない。

平面スクリーン映像の視覚特性は以下である。

1. 標準視角 40^0では、変形しにくい。
2. 視角 23^0では、肉眼視覚に近く極めて自然な視覚効果表現が得られる。

計算により 50^0くらいの視角の場合、ボリューム中部空間を通過する視線の密度が周辺と同じみてよい、故に,最小 54^0に制限された視角でさえ、レイキャスティングを行うと,科学的な可視化の要求が十分に満足できることが明らかである。

以降、本稿に述べたピクセル並列処理は視角制限のピクセル並列処理を指すことである。

3.2 ピクセル並列処理の性質

3.2.1 ボリューム空間からメモリ空間への写像

ボリューム空間を、図 1のように、L 軸の整数座標点 $1,2,\ldots,n-1$ に沿って, L 軸に垂直な n 個のスライス $L_i(i=0\sim n-1)$ に薄く切り、L 軸上のスライス群とする。L 軸上の i $(i=0\sim n-1)$ 番目スライス L_iの厚さは、該当軸上の半開区間 $[i,i+1)$ に対応する。ここで、各スライスの横断面はスライスの断面、スライスの厚さを表す面はスライスの側面、スライス群の両側にある L 軸に垂直な表面はスライス群の底面、スライス群のあと 4 つの表面はスライス群の側面と名付ける。

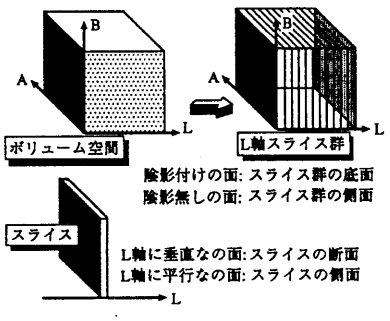

図 1: L 軸に沿ったボリューム空間の分割

こうすれば、ボリューム空間はスライス群に対応したメモリバンク群に格納できる。

後述では、断らない限り、スライス群というのは、メモリバンク群に格納されたボリューム空間を指すことである。

3.2.2 スライス群を囲んだ視点空間の区分

視点からスクリーンを通してスライス群空間を見る場合、スライス群二底面のスクリーンへの投影が重なる視点空間はスライス群の底面観察空間、重なっていない視点空間はスライス群の側面観察空間と呼ぶ。

図 2 に示すようにスライス群の底面観察空間はスライス群の中心を通る四つの対角線の延長線で限られた、スライス群の底面を含む上下 2 つの陰影囲いの四角錐空間である。それ以外の空間はスライス群の側面観察空間である。

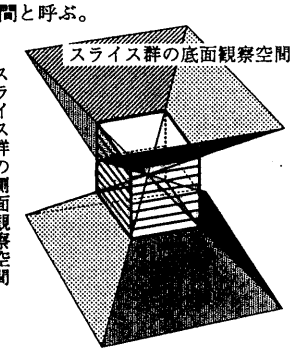

図 2:視点空間の区分

明らかに、視点がスライス群の側面観察空間にある場合、二底面を通る視線がない。さもないと、視点がスライス群の底面観察空間にある。

3.2.3 ボリューム空間への全視線入射規則

全ボリューム空間を観察するため、視線の入射は以下の 2 つの規則に従う。

規則 1 ボリューム空間の全てを全スクリーンの n^2本視線からなる視線空間の囲いに取り入れ、スクリーンの中に投影する。

規則 2 視点からスクリーン中心を通過する方向線を、終始一貫,ボリューム空間の中心へ指向させ、ボリューム空間はスクリーン上の投影面積が最大となるようにする。

3.2.4 ピクセル並列処理の前提

視角制限方式のピクセル並列処理では、以下の点を前提としている。

前提 1 視点からボリューム中心までの距離: スクリーンを通る n^2本の視線と,視点から見えるボリューム表面との交点のうち、スライス群の側面にある、L 軸に沿った隣接交点の間隔がボクセル稜線の長さより短くないように決める。これで、ボリュームに代わったスライス群の側面上のボクセルに高々 1 本の視線しか到達できない。

前提 2 スクリーン上の視線群選択: 視点からスクリーンを通して、座標系 O_{LAB}に定義したスライス群を見る時に、スライス群の側面か底面か底面と側面か底面と二側面が見える場合にしか限らない。

(a) もしスライス群の一側面しか見えないならば、その側面が LA(または LB) 平面に必ず平行である。仮に A

軸 (または B軸) のスクリーンへの投影がスクリーンの行線となす角θは45^0以上であるならば、スクリーンの1行分ピクセルを通るn本視線を視線群とする。さもないと、1列分ピクセルのn本視線を視線群とする。

(b) もしスライス群の二側面が見えるならば、ただ一つだけの側面が見えるようにするため次のスライス群幅拡張規則によって各スライスの幅を広げる。

スライス群の幅拡張規則：B軸とスクリーンとのなす角は、A軸とスクリーンとのなす角より
- 大きいならば、各スライスをA軸に沿って伸ばす。
- 等しいならば、AとBのいずれかに沿って伸ばす。
- 小さいならば、B軸に沿って伸ばす。

ただし、伸ばされた部分はボクセルの透明度を1、色を0にする。こうすれば、**前提1**により視線群の各視線とスライス群の側面との交点がいつも各スライスの側面毎に1個を超えないようになっている。

前提3 全スクリーン分視線のスライス群への入射方法：視線群ずつ投入する。

前提4 視線群のスライス群への入射方法：
1. スライス群の**側面**に至った視線を、**同時**に入射する；
2. スライス群の**底面**に至った視線を、**順番**に入射する。

前提5 視線がボクセルを通過する時間間隔：それ(ボクセル値の処理時間) をΔTにする。

3.2.5 ピクセル並列処理の性質

3.1節の視角制限方式下にはスライス群を通過するピクセル並列処理による視線群が以下の性質を満たすことが証明できる。

性質1 異なる視線が同時に同一スライスを通過することはない。

性質2 もし視点がスライス群の側面観察空間にあるならば, 視線群は$n \times \Delta T$ の始点にスライス群空間に進入し始め, 必ず、その$n \times \Delta T$ までスライス群を出て来る。

性質3 もし視点がスライス群の底面観察空間にあるならば, $n \times \Delta T$ ごとに新しい視線群がスライス群に進入でき、また各視線群がスライス群を通る時間は$2 \times n \times \Delta T$を超えない。

以上の各性質の証明は参考文献 [1] を見て頂く。

3.3 バンクコンフリクト無しのメモリ構成

スライス群の各スライスが各々メモリバンク群に対応しているので、3.2.4の**前提5**と3.2.5の**性質1**によって、ピクセル並列処理を行っている視線群の各視線が同じΔTで同一メモリバンクをアクセスすることはないということが分かった。故に、各視線は、同一ΔTで各自が通過しているところのメモリバンクからボクセル値を読み出して、2.4.1節の漸化式によるレイキャスティングを行っていく。

3.4 マクロ・パイプラインによるPixel並列処理

3.4.1 視線群特徴のまとめ

3.2.5節の**性質2**と**性質3**をまとめてみると、以下の特徴がある。
(1). 視線群が必ず$n \times \Delta T$でスライス群の中に進入できる。
(2). 視線群が多くても$2 \times n \times \Delta T$までスライス群を出る。
(3). 複数の視線群がスライス群の内に$n \times \Delta T$に視線群ずつ進入できる。

これにより、1枚の画像生成は、全スクリーンを通るn個視線群が順次に以下3.4.2節のステージに$n \times \Delta T$のマクロ・パイプラインに流れ込むことで行う。更に連続的な画像生成にも、このマクロ・パイプラインにより停滞せずに行える。

3.4.2 マクロ・パイプラインの導入

一枚の画像に関するn個視線群が順番にスライス群に入射する処理は、図3に示したマクロ・パイプラインで行う。

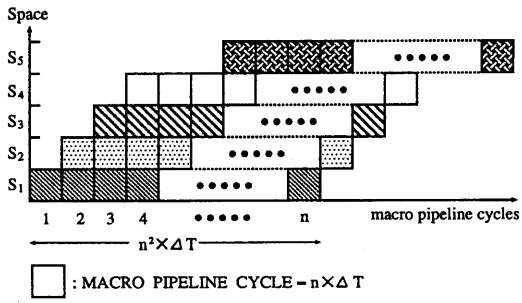

図 3: 1枚の画像生成の視線群マクロパイプライン

S_1： 視線群計算ステージ：$n \times \Delta T$ で視線群のスライス群へのn個入射点を算出する。

S_2: 入射点ディスパッチステージ:$n \times \Delta T$ で計算済のn個の入射点パラメータをスライス群表面の対応位置にΔTに1個ずつ送出する。

S_3： ピクセル並列レイキャスティングステージ：$n \times \Delta T$で視線群の各視線をスライス群の内に入射済みさせ、また、進入した視線がレイキャスティングを行っていく。

S_4： 色バッファとZバッファ構築ステージ：$n \times \Delta T$ で、スライス群から出てきた各視線のピクセル値をΔT にピクセル値ずつ色バッファの該当ピクセルの位置に書き込むとともに視線がある表面に交差するまで視線に沿って進んだ前進ステップの数によって求めたピクセルの深さ値をZバッファに書き込む。

S_5： シェーディングステージ:$n \times \Delta T$ で、周辺のピクセルの深さを参照して表面の法線を近似に計算し、それをもとに各ピクセルの輝度を求める depth gradient shading 法により、視線群各視線のピクセルの輝度計算をΔTにピクセルずつ逐一に完成する。

図3のS_5では、スクリーンの512×3個ピクセルのdepth値を形成してから始めて、近傍ピクセルの深さを参照するシェーディングを行う。

明らかに、$n \times (n \times \Delta T)$で全スクリーンを通る$n^2$本視線がスライス群空間に全部に入射できるので、パイプラインにより$n^2\Delta T$ごとに1枚の画像は形成できる。

3.5 マクロ・パイプライン構成の概要

図4は3.4.2節のマクロ・パイプライン構成の概要である。その中に主としてメモリバンク群を中心としたピクセル並列レイキャスティングステージを図示した。

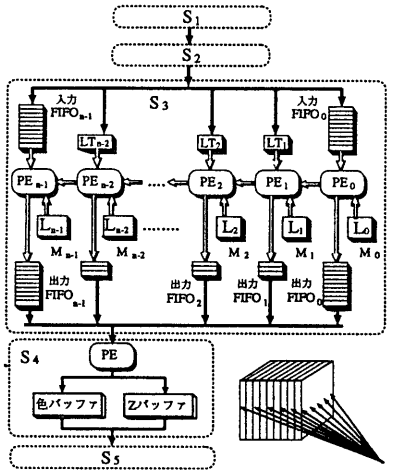

図4: メモリ構成を中心としたマクロパイプライン

漸化式表示のレイキャスティングによる累積カラー計算の特徴としては、視線群の各視線に沿った累積カラー計算を行っているところの各ボクセルは、そのボクセルが視線に沿って前後に隣接したボクセルでしか必要な計算を行っていない, という特徴である。

この特徴によって、メモリバンク群を中心としたピクセル並列レイキャスティング・ステージでは、同一構造をしたn個プロセッサをメモリバンクに1個ずつ1次元アレイ状に規則正しく配置し、直接隣接したプロセッサしか1ステップ(単位時間)で直接通信できないような構成を採用している。

入射点ディスパッチステージでのアレイへのn個視線入射点の入力と、色バッファとZバッファ構築ステージでの計算済みのn本視線ピクセル値の色バッファとZバッファへの出力を逐次に行い、メモリバンク群に進入した各視線は、各々各自が通過しているところのメモリバンクに配置したプロセッサで、2.4.1節のレイキャスティングの漸化式(3)に従って、一つの前進ステップの計算(2.4.3節)を並列に行い、しかも、各視線に沿った累積カラー計算は、同時に、各視線が順番に通っている各メモリバンク上のプロセッサに沿ってパイプライン方式で重畳的に進めていく。

各プロセッサで行っている視線の前進ステップの計算時間間隔が同じなので、直接隣接のプロセッサへの移行に必要な各視線について移行するためのデータ転送は必ず同時に行う。故に、データ転送に必要な各プロセッサは隣接プロセッサへのデータ転送を同期に動作している。

一方、図4の中に視線群が右側からスライス群に入射するとすると、スライス群の底面に到る視線の入射点がPE_0の入力$FIFO_0$に、スライス群の各スライス側面に到る入射点が、対応したプロセッサの入力ラッチに格納されるようになっている。よって、スライス群の側面に到る視線は同時に入射でき、スライス群の底面に到る視線は順番に入射できる。故に、3.2.4節の前提4という視線群の入射方法が満足される。

また、その視線群がスライス群を出た時、各視線の累積カラーの最終値, ピクセル値は、スライス群の底面から出たものが底面に対応なPE_{n-1}の出力$FIFO_{n-1}$に、各スライスの側面から出たものが、そのスライスに対応するPEの小さな出力$FIFO$に書き込まれている。

3.6 入出力FIFOに決る視線群各視線の進出順

3.6.1 色バッファとZバッファへの色とZ値の書き込み

スクリーン座標点(X,Y)のピクセルを通った視線がスライス群表面の点(L_{in}, A_{in}, B_{in})に入射して、それから、スライス群表面の点$(L_{out}, A_{out}, B_{out})$から出てきた。その視線に沿って生成した色及びZ値は、色バッファ及びZバッファの(X,Y)位置に書き込まなければならない。言わば、その視線の色及びZ値が$\lceil L_{out} \rceil$††番のプロセッサの出力FIFOから取り出して色バッファ及びZバッファの(X,Y)位置に書き込むということである。ここで、$\lceil L_{out} \rceil (\lceil L_{out} \rceil = 0, 1, \ldots, n-1)$は、L軸に垂直なn個スライスの番号である。

3.6.2 実現方法

簡単な方法としては、(X,Y)を、視線に沿う累積カラーの計算に伴って転送し、最後、色とZ値及び(X,Y)が$\lceil L_{out} \rceil$番プロセッサの出力FIFOに書き込まれることにより、色バッファとZバッファ構築プロセッサが出力FIFOからそれを取り出して色及びZ値を色バッファとZバッファの(X,Y)に書き込む。

ところが、この方法は転送配線数の増加に導こうとする。これを回避するため、以下方法を採用する。

入射点ディスパッチステージと, 色バッファとZバッファ構築ステージとの間にn以上長さの$FIFO_S$で直結する。視線の$\lceil L_{out} \rceil$を入射点と一緒に算出する。

入射点ディスパッチステージでは、入射点をスライス群表面に送出するとともに、その視線の$(X,Y), \lceil L_{out} \rceil$を$FIFO_s$に書き込む。

色バッファとZバッファ構築ステージでは、$FIFO_S$から$(X,Y), \lceil L_{out} \rceil$を取り出し、$\lceil L_{out} \rceil$を用いて、ピクセル並列レイキャスティングステージの$\lceil L_{out} \rceil$番のプロセッサの出力FIFOからいろとZ値を取り出して、色バッファとZバッファの(X,Y)位置に書き込む。

ピクセル並列レイキャスティングステージの各出力FIFO内のピクセル値の配列順を$FIFO_S$内の$(X,Y), \lceil L_{out} \rceil$の

†† $\lceil X \rceil$: Xを超えない最大の整数

配列順に一致するため、入射点ディスパッチステージでは、以下の入射点のディスパッチス順序に従い、入射点と$(X,Y),\lceil L_{out}\rceil$をレイキャスティングステージに送出する。

3.6.3 $(X,Y),\lceil L_{out}\rceil$を含む入射点のディスパッチ順序

視線群の各視線$R_i(i=1\sim n)$からなる平面上、視点からスライス群の断面に平行な直線r_Aを引く。入射点のディスパッチ順序は以下である。

1. もし視線群がr_Aの片側にあるならば、r_Aとなす角の大きい視線の入射点が優先ディスパッチされる。
2. もしr_AがR_jと$R_{j+1}(1\leq j\leq n-1)$の間を通過すると、入射点のディスパッチは、
   $R_1,R_2,\ldots,R_j,R_n,R_{n-1},\ldots,R_{j+1}$の順である。

4 本計算機の構成

以下本稿では、ボリューム空間、スクリーンの大きさ、ΔTは、各々$512^3 voxel$, $512^2 Pixel$, $80ns$であるとする。

4.1 全体構成

3章の討論に基づくボリュームレンダリング向きの並列計算機の概念構成を図5に示す。構成上の特徴としては、
(1). システムコントロールユニットが、ホストマシンからの指示に従って、$512\Delta T$にステージの、ピクセル並列処理によるレイキャスティングを行う5ステージマクロパイプラインを制御することである。従って、$512\Delta T$ごとにスクリーン1行分(または1列分)の512本視線を、このマクロパイプラインに書き込むことにより、$512^2\Delta T \doteqdot 21ms$に1枚画像、1秒間に48枚画像の作成が実現される。
(2). 多様な処理に対して柔軟に対応可能な構成となるため、書き換え可能なゲードアレイ(FPGA: Field Programmable Gate Array)を用いることで再構成可能なアーキテクチャが達成される, ということである。

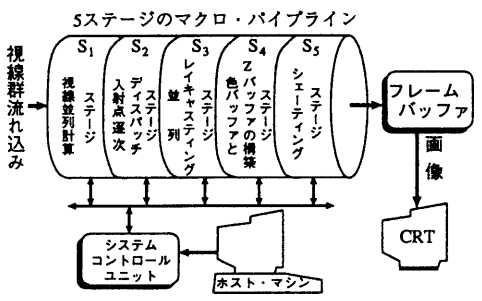

図 5: 本計算機の概念構成

4.2 SCU (システムコントロールユニット)

SCUは、システム全体の初期設定及び制御を行う機能ユニットであり、TMS320C40 1プロセッサで構成される。初期設定時においては、図5のS_1ステージの32個DSPにプログラムを、S_1,S_2,S_3,S_4ステージのFPGAで構成されるプロセッサに内部構造データを転送する。また、ボクセル色/透明度テーブル、動作モード、視点とスクリーンの位置関係などの各種パラメーターを各構成要素に指定する。

4.3 マクロパイプライン

本計算機の全体構成は図6に示す。

4.3.1 視線並列計算ステージ

図6のstage1に示すように、32個TMS320C40 DSPが並列に配置されている。

このステージでは、$512\Delta T$に512本視線の入射点パラメーター:$(X,Y),\lceil L_{out}\rceil,(L_{in},A_{in},B_{in}),\Delta D,(\Delta L,\Delta A,\Delta B)$などを計算して、各DSPの出力FIFOに書き込む。512本視線の32個DSPへの割り付けは、DSP_iの出力$FIFO_i$から$i=0,\ldots,31$の順に入射点を取り出し、それを繰り返して、512個入射点が取り出され、配列する時、3.6.3節の入射点ディスパッチ順序を満たすようにする。

4.3.2 入射点ディスパッチステージ

このステージでは、1チップの$FPGA-PE_{RC}$を配置する。上に述べたように、32個DSPの出力FIFOから順番に512個入射点パラメータをΔTに1点ずつ取り出してから、ピクセル並列レイキャスティングステージの対応する入力ラッチあるいは入力FIFOに、$512\Delta T$に512本視線の入射点パラメーター:$(L_{in},A_{in},B_{in}),\Delta D,(\Delta L,\Delta A,\Delta B)$を書き込み、色バッファとZバッファ構築ステージに連結する$FIFO_S$に同じ入射点の$(X,Y),\lceil L_{out}\rceil$を書き込んでいく。

4.3.3 ピクセル並列レイキャスティングステージ

図6のstage3に示すように、パイプライン化されたFPGA-$PE_i(i=0,\ldots,511)$が1次元に接続された構造であり、各PE_iは、ボリュームのスライス$_i$のデータ、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。PE_0とPE_{511}が入力FIFOから入射点を順番に、$PE_i(i=1,\ldots,510)$が各自の入力ラッチから入射点を同時に取り出して、並列レイキャスティングを行う。

$PE_i(i=0,\ldots,511)$が$(\Delta T=)80ns$に1ステージのmステージのパイプライン構成に設計される(離散モデル:m=5, 連続モデル:m=7)。このmステージのパイプライン$PE_i(i=0,\ldots,511)$により、各々離散モデル:$5\Delta T$, 連続モデル:$7\Delta T$で1前進ステップの計算(2.4.3節参照)を完成する。また、80nsごとに新たな前進ステップの計算を始め、80nsごとに1前進ステップの計算を完成する。$PE_i(i=0,\ldots,511)$が同期に動作しているので、各視線が隣接のPEに移行するデータ転送も同期に行う。視線が$\lceil L_{out}\rceil$番目のスライスから出た場合、色と、視線がある表面に交差するまでのdepth値とが、$PE_{\lceil L_{out}\rceil}$の出力FIFOに書き込まれる。

時間軸に沿う動画作成については、図6のstage3に示すように、PE_iに2つのメモリバンクずつ$(i=0,\ldots,511)$を配置して、その一つはボリュームレンダリングを行うと同時にもう一つはデータ更新を行うことができるようにする。

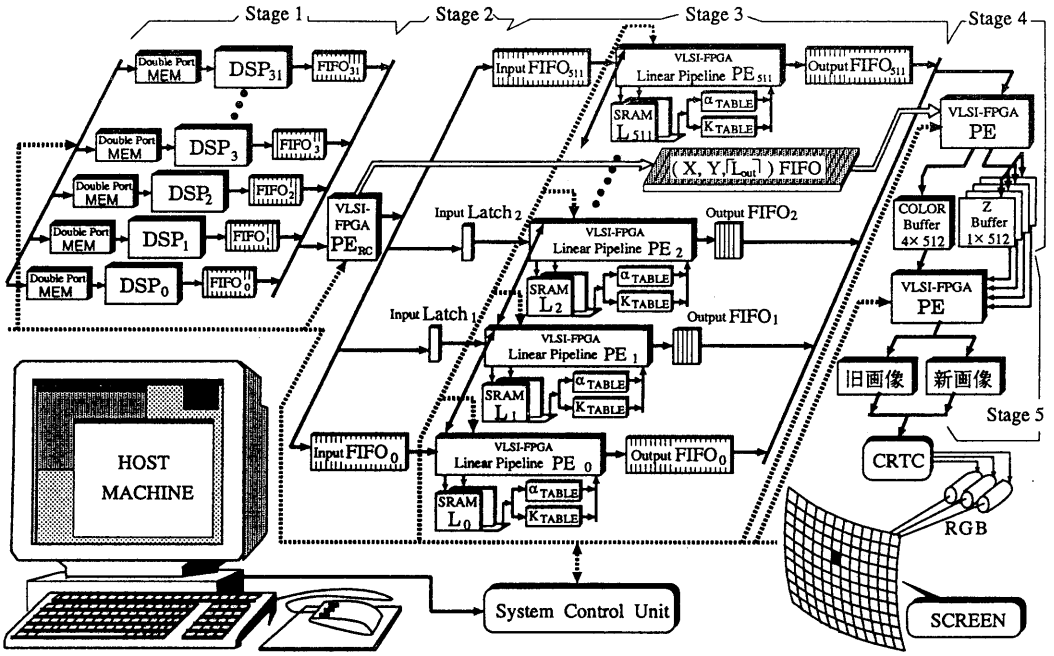

図 6: 本計算機のアーキテクチャ

4.3.4 色バッファとZバッファ構築ステージ

ここでは、1 FPGA − PE が配置される（図6の stage4 参照）。この PE は、入射点ディスパッチステージに連結した $FIFO_S$ から (X,Y), $\lceil L_{out} \rceil$ を ΔT に1個ずつ取り出し、$\lceil L_{out} \rceil$ によって、ピクセル並列レイキャスティングステージの出力 FIFO から色と Z 値を取り出して、各々、色バッファと Z バッファの (X,Y) 位置に書き込む。

4.3.5 シェーディングステージ

図6の stage5 に示すように、ステージに ΔT の7ステージのパイプライン FPGA-PE が配置される。色バッファと Z バッファ構築ステージによって 3×512 個色と Z 値を色バッファと Z バッファに書き込んだ後、この PE は、3×3 ピクセルの Z 値上に、その中心にあるピクセルの色に depth gradient shading を施す。パイプライン化された PE なので、$\Delta T = 80ns$ ごとにシェーディングされたピクセル値が作成できる。

4.4 連続モデルでのメモリ構成

本計算機では、ボクセルの八頂点の tei-linear 補間に代わって、四頂点補間を行う。重畳交錯の3重化メモリを利用し、2回アクセスだけで四頂点を取り出しできる。離散モデルでは、同じスピードでボリュームレンダリングが行える。紙の都合で詳細の説明は省略させて頂く。

5 おわりに

ボリューム表示を超高速に行うアーキテクチャを提案した。その特徴は,1) 半透明表示,2) 透視投影,3) 主軸等間隔サンプリングによるレイキャスティングの採用, である。このアーキテクチャによるボリューム表示の速度について、n を 512, 固定パイプラインサイクル ΔT を 80ns とするので、画面生成率は 48 画像/秒である。ピクセル値計算の各プロセッサが $m\Delta T$ で一つの前進ステップ (ΔD 間隔) 累積カラーの計算を完成すると, 各プロセッサを m ステージのパイプラインに設計して,$\Delta T = 80ns$ ごとに累積カラー値を流させてくる。これらのプロセッサはプログラマブル VL SI-FPGA で設計する。

謝辞

日頃から御討論頂く京都大学工学部富田研究室の皆様に感謝致します。

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のアーキテクチャ, 情報処理学会計算機アーキテクチャ研究会, 95-ARC-13, 1995.8

[2] Hanspeter pfister. Cube-3: A Real-Time Architecture for High-resolution Volume Visualization. In proceeding of 1994 Workshop on Volume Visualization