修士論文

大規模非構造格子データの並列可視化における動的負荷分散

指導教官　　富田 眞治 教授

京都大学大学院情報学研究科
修士課程通信情報システム専攻

高山 征大

平成16年2月6日

大規模非構造格子データの並列可視化における動的負荷分散

高山 征大

内容梗概

近年の汎用 PC の性能向上はめざましいものがあり，これを並列に連携させることで大きな処理能力を得る PC クラスタの構築が盛んである．こうした PC クラスタの用途として，数値シミュレーションがある．そして，数値シミュレーションの結果を，人間に理解できるように数値から画像に変換する作業のことを可視化といい，本研究では，これら数値シミュレーションとその結果の可視化とをひとつの PC クラスタで行うシステムの構築を目指している．

このようなシステムにおいては，シミュレーションが生成するデータは個々のプロセッサに分散して出力される．そして，その配置は可視化にとって必ずしも最適なものとは限らず，結果として負荷の非均衡などの問題をもたらす．しかし，これまでにこのデータの配置に自由度がない状況での可視化について論じられることはなかった．

本稿では，この問題を解決するために動的負荷分散を用いた場合の考察を行った．いくつかの動的負荷分散方針を検討し，評価を行ったところ，負荷の均等化を達成し，全体の処理時間の削減に成功した．

また，刻々とデータを生成する数値シミュレーションに応じて実時間で画像を出力できるシステムの構築を目指し，可視化を高速化するために Early Ray Termination(ERT) 技術に関する考察も行った．本研究で採用する可視化アルゴリズムである Cell Projection 法において，ERT を行った例はない．本研究では，Cell Projection での ERT 手法である ERT-table 法を提案し，評価を行った．その結果，最良の場合において 3.86 倍の性能向上を達成した．

さらに，並列可視化における ERT についても考察を行い，ERT 情報を共有させることによって高速化を図る手法を提案し，評価を行った．その結果，1.9 倍の性能向上を達成した．

そして，以上で提案した手法を全て組み合わせた場合において，提案手法を用いない場合と比較して 4.32 倍の性能向上を達成した．

Dynamic Load Balancing on Parallel Visualization of Large Scale Unstructured Grid

Motohiro TAKAYAMA

Abstract

There is the remarkable improvement in performance of general-purpose PC in recent years, and the construction of the PC cluster is getting popularity because it obtains a big processing performance by cooperating PCs in parallel. There is a numerical simulation as a usage of such a PC cluster. And, the work converted from the numerical value into the image so that man may understand the result of the numerical simulation is called visualization, and this research aims at the construction of the system which does both numerical simulations and visualization of its result with one PC cluster.

The data which the simulation generates distributes to an individual processor and is output in such a system. And, this arrangement does not always fit for visualization, and bring the problem of load-imbalance etc. However, this problem has not been discussed yet. In this paper, we considerd for using dynamic load-balancing to solve this problem. Some dynamic load-balancing policies were examined, and evaluated. As a result, the good load-balancing was achieved, so we succeeded in the reduction in the entire processing time.

Moreover, to aim at the construction of the system which can output the image in real time according to the numerical simulation by which data is second by second generated, and to speed up visualization, we considered the Early Ray Termination(ERT) technology. In the Cell Projection which is the visualization algorithm adopted by this research, there is no example of having done ERT. In this research, the ERT-table method which was the ERT technique in Cell Projection was proposed, and evaluated. As a result, 3.86 performance times improved were achieved in case of the best. In addition, the technique to speed up parallel visualization by sharing ERT information was proposed, and evaluated. As a result, 1.9 performance times improved were achieved. And, when all the techniques proposed above were combined, 4.32 performance times improved were achieved compared with the case where the proposal technique is not used.

大規模非構造格子データの並列可視化における動的負荷分散

目次

第1章　はじめに

近年，汎用 PC の性能向上はめざましいものがあり，かつてはスーパーコンピュータを必要としたような演算も，安価な PC で行えるようになってきている．そして，こうした PC を複数台ネットワークで結ぶことにより，大きな計算能力を得ようという PC クラスタの構築が盛んである．

PC クラスタによる並列計算機は，従来のスーパーコンピュータと比較した場合，ハードウェア，ソフトウェア共に圧倒的に安価であるという廉価性，計算機を構成する部品を調達する際に容易に入手可能である可用性，小組織でも構築できることにより，計算能力を欲するときに利用できるという専有性，などの点で優れている．

こうした PC クラスタの用途の一つとして，大規模かつ高精度な数値シミュレーションがある．そして，そのシミュレーション結果を，人間に理解しやすいよう画像にして提示する可視化技術は，数値シミュレーション結果の意味を理解し，また次のシミュレーションを方向づけるための一助となる役割を持つという点で，非常に重要であり，これまでに多くの研究が行われている．

中でも実時間の数値シミュレーションの可視化技術は，大規模な三次元データを必要とする医療などの分野 [1] において，高度な技術が要求されている分野である．これまでにも，並列計算機を利用した，シミュレーションとその実時間可視化のためのシステムが開発されてきたが，次世代のシミュレーション技術として，従来の実験の代替手段となりうる「仮想実験型/仮想体験型のシミュレーション環境」の構築が望まれている．ここでは，オペレータによるシミュレーション対象へのインタラクティブな操作に対応して，実時間でシミュレーションを行うとともに，即時にその結果を可視化などの手段により提示することが求められている．

そのような実時間インタラクティブシミュレーション環境の実現にむけて，シミュレーションと可視化，そして両者の連携について研究を行っている．当面は，4096^3 規模のデータを，SHD 規格相当のスクリーン (2048^2) に，秒間 30 枚のフレームレートで出力する可視化システムを構築することを，目標とする．可視化手法としては，シミュレーション結果を等値面に変換する Surface Rendering ではなく，数値を色情報に変換する Volume Rendering を用いる．

並列 Volume Rendering 環境の実現方法として，3 つの方針で研究を行ってい

る．まず，処理対象に応じて処理内容を柔軟に変更でき，処理の高機能化も容易であるソフトウェアによる方法．次に，昨今CPUよりも性能向上が顕著であり，今後の発展が期待できるGPUによる方法．最後に超高速化を目指した専用ハードウェアによる方法である．

本稿は，これら3つの方針のうちの，ソフトウェアによる方法について述べるものである．対象とするデータの種類は，数値シミュレーションとして一般的な有限要素法などが生成する非構造格子とし，データの規模は並列化が不可欠である程度とする．そして，非構造格子データをPCクラスタで並列に可視化する場合において生じる問題について述べ，その問題を解決するための動的負荷分散技術について述べる．また，高速化技法として， Early Ray Temination 手法，Binary Swap Composition 手法についても述べる．そして，PCクラスタ上に実装した各手法の評価，考察を行う．

第2章　背景

本章では，本研究の背景となる諸要素について述べる．まず，数値シミュレーションとその結果の可視化，特にVolume Rendering法について概観し，両者を同じ分散メモリ型並列計算機内で行うときに生じる問題を指摘する．次に，本研究で扱う非構造格子の性質を説明し，非構造格子を可視化する方法の一つであるCell Projection法について述べ，その並列化手法，合成についても言及する．また，第3章の背景として，Early Ray Termination手法についても述べる．最後にこれらをまとめた場合に生じる，負荷の非均衡の問題を指摘する．

2.1　数値シミュレーションの可視化

数値シミュレーションの結果である数値データを可視化する方法の一つとして，Volume Renderingがある．Volume Renderingとは，与えられた三次元データを別形状のデータに変換することなく，そのままの形状で数値から透明度を伴う色に変換を行い，半透明状の画像を得る手法である．データを等値面に変換して可視化を行うSurface Renderingでは，物体の表面だけを可視化対象としているのに対し，Volume Renderingでは物体の中が詰まったまま可視化処理を行うため，具体的な形をとらないガス状に分布するシミュレーション結果の可視化や，物体の内側を観察することが必要となる医療シミュレーション結果の可視化などにおいて，有効な可視化手法である．数値データは，なんらかの計算式によって得られるスカラ値，あるいはベクトル値であり，以降この値のことを関数値と呼ぶ．また，関数値から色情報を得るのに用いる関数を伝達関数と呼ぶ．この関数を任意に変更することによって，ある関数値範囲にあるデータだけを観察したいといった要求に応えることが可能である．

各ピクセルでの色値は，次式によって計算できる．

$$I = \int_0^D c(v(\lambda)) \exp(-\int_0^\lambda \tau(v(\lambda'))d\lambda')d\lambda \tag{1}$$

Iは求める色の輝度，λは空間上での位置，Dはデータの存在する位置の上限，$v(\lambda)$は位置λでの入力関数値，$c()$は伝達関数で求まる色成分，$\tau()$は伝達関数で求まる不透明度成分，である．

この積分式を計算機で扱えるよう離散化すると，次の式が得られる．

$$I = \sum_{i=0}^{n} \alpha(v_i)c(v_i) \prod_{j=0}^{i-1}(1 - \alpha(v_j)) \tag{2}$$

λ は i に，n は D に，$\alpha()$ は $\tau()$ に，それぞれ相当する．

この畳み込み演算は，演算区間をいくつかの部分区間に分割し，それぞれの区間に対する計算結果に対して，再度畳み込み演算を行うことで結果を得ることが可能である．すなわち，Volume Rendering はデータ並列性が高く，並列化することによって容易に高速化が期待できるという性質がある．

Volume Rendering のアルゴリズムは，処理の方向によって大きく二つに分類することができる．一つは，後方投影法といい，スクリーン上のピクセルごとに処理を行っていくもので，代表的なものが Ray Casting 法 [2] である．Ray Casting 法では，まず視点とスクリーン上の各ピクセルとを結ぶ光線をデータ (オブジェクト) に投げかける．次に，光線がオブジェクトを通過する際に，補完によってその点での関数値を求める (サンプリング)．そして，(2) 式を用いて，現在の位置での色情報と不透明度を計算する．以上の処理を，光線がオブジェクトを通り過ぎるまで続けることにより，最終的に 1 ピクセルの色情報を得る．これを，スクリーン上の各ピクセルに対して行うことで，最終画像を得る．

もう一つは前方投影法といい，オブジェクト空間で可視化処理を行った結果を，スクリーンに投影するという順で処理を行う．この方法には，Cell Projection[3] 法，Shear-Warp[4] 法，Splatting 法 [5]，Texture-based 法，Scanline 法，Sweep 法など，数多くのアルゴリズムが発表されている．

本研究では，非構造格子を対象としていること，またデータの配置に対する自由度が高いことから，前方投影法の一つである Cell Projection 法を採用する．

2.2 データの配置に自由度がない状況での可視化

いま，数値シミュレーションが PC クラスタのような分散メモリ型並列計算機で並列に実行され，その結果を可視化処理する場合を考える．数値シミュレーションとして一般的に用いられる有限要素法では，計算量は一般に要素数に比例する．そのため，数値シミュレーションでは，要素数をプロセッサ間で均等にするような静的負荷分散方針が有効である．

ところが，数値シミュレーションにとって効率の良いデータ分散が，必ずしも可視化処理にとって効率の良いデータ分散であるとは限らない．第一に，要素数がプロセッサ間で均等であっても，可視化処理における負荷が均等であるとは限らない．第二に，可視化するデータの個々の格子 (セル) が空間的にどのような分布であるかということが，可視化に与える影響が大きい点である．Ray

Casting 法のように，可視化アルゴリズムによっては隣接するセルに対するアクセスが頻繁に発生する．一方で，数値シミュレーションでは負荷分散を図るため，実際には幅 w のブロックサイクリック分割を行うことが普通であり，そのような場合に，可視化側は隣接セルを処理する w 回毎にプロセッサ間通信をする必要があり，効率が悪くなる．

このように，シミュレーション結果として生成されたデータを，そのままの分散配置で可視化しようとすると，負荷の非均衡，通信量，通信頻度の増大によって，全体の処理時間が長くなってしまい，第 1 章で求められているようなインタラクティブ性を実現できない．PC クラスタのようにネットワークが低速である分散メモリ型並列計算機においては，可視化処理を行う前に可視化にとって効率の良いようにデータを再分配する方法をとると，データ量の大きさ故に通信のオーバーヘッドが大きくなり，結果としてインタラクティブ性を実現できない．

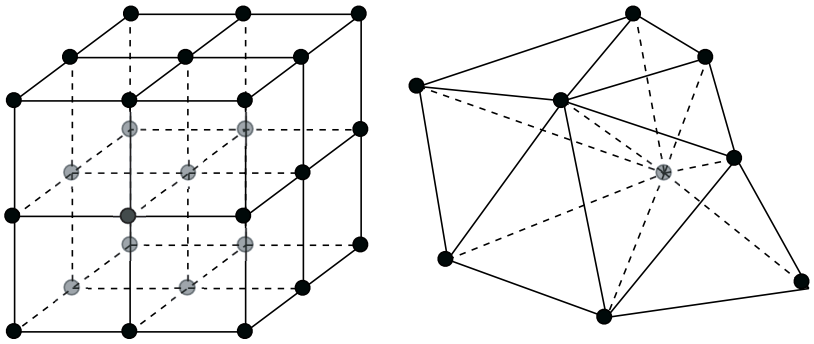
これまでの並列 Volume Rendering アルゴリズムでは，このデータの配置に自由度がない問題について考慮されることがなかった．例えば，シミュレーションとその可視化を行う同様のシステムとして，GeoFEM 可視化サブシステム [6] があるが，これは共有メモリ型並列計算機上での実装であるため，上述のような問題は起こらない．他の同様のシステムである VGcluster[7] では，データの領域分割を静的に行うことにより，負荷の非均衡の解決を図っている．また，分散メモリ型並列計算機で並列 Volume Rendering を効率よく行うアルゴリズム [8] も提案されているが，初期配置としてデータを自由に分配できることが前提となっている．

2.3 非構造格子データ

Volume Rendering が扱う三次元データは，その要素である格子の形状や並び方によって二種類に大別できる．一つは，個々の格子が規則的に並んでいる構造格子であり，例えば MRI や CT スキャンなどの結果として得られる (図 1 (A))．もう一つは，個々の格子の並び方が不規則であり，大きさ，形状も非均質な非構造格子であり，有限要素法などの重要な数値シミュレーションによって生成される (図 1 (B))．

それぞれのデータ形式には，次のような得失がある．

データ量



(A) 構造格子　　(B) 非構造格子

図 1: 各格子

一つの格子を決定づけるデータとは，座標値と格子を囲む頂点の関数値である．構造格子においては，任意の格子の座標値は，データ全体の端点の座標値から，単純な計算によって求めることができる．したがって，個々の格子が座標値を持つ必要はなく，データ全体における一頂点の座標値を持つだけでよい．それに対し，非構造格子では，隣接する頂点に幾何学的構造が存在しないため，計算によって隣接する頂点の座標値を得ることができない．そのため，各頂点が座標値を持つ必要がある．また，格子の形状が自由であるため，格子を構成する頂点配列，格子がどのような形状であるかの情報も必要である．

したがって，非構造格子は構造格子よりも多くのデータ量を必要とする．

形，大きさ

構造格子は，形も大きさも均質であるため，個々の格子を処理するのに単純な反復しか要さない．一方，非構造格子はそれぞれの格子が自由な形と大きさをとることが可能なので，個々の格子の処理が複雑になる．

しかし，非構造格子はその自由度によって，任意の形状を表現することが可能である．それに対し，構造格子で複雑な形状を表現するには，多くの小さな格子を用いる必要があり，データ量が増大するという欠点がある．

ソート

ある空間軸方向に対するソートを考えるとき，構造格子では，個々の格子は構造上既にソート済みである．その一方で，非構造格子では多様な形状があるため，ソートを行うのが困難である．また，格子の形状によっては，ソートが不可能な場合もある．例えば，四面体から成る非構造格子 $cell_1, cell_2$

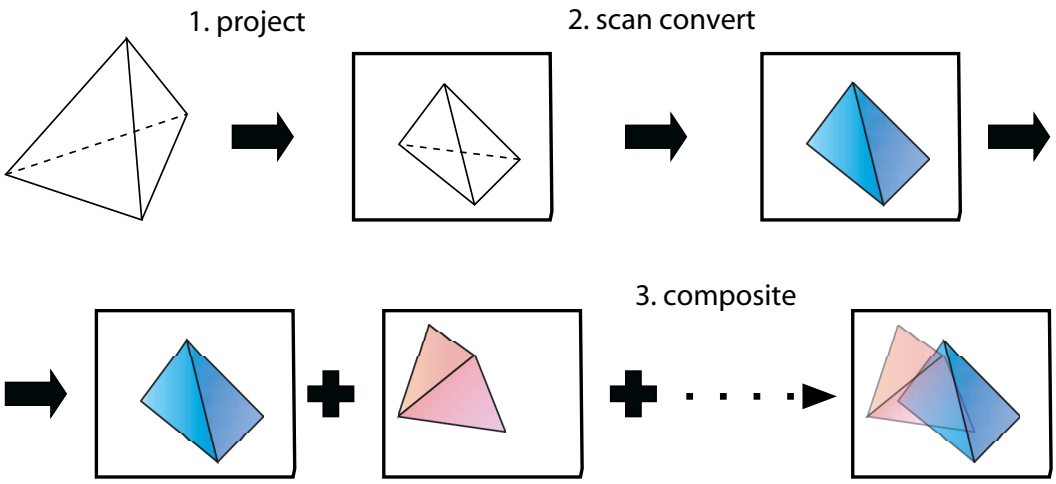


図 2: Cell Projection

を，ある方向から投影した投影面について考える．このとき，投影面に含まれる点 P_a では $cell_1$ が手前にある一方で，また別の点 P_b では，$cell_2$ が手前になる，という場合が存在する．

以上のように，非構造格子は扱うのが困難であるが，形状の表現力に優れているため，数値シミュレーションでよく用いられている．前処理によって，非構造格子を構造格子に変換した後に Volume Rendering を行う方法も提案されているが，構造格子への変換は容易なものではないため，前処理に要するオーバーヘッドが問題となる．本研究では，数値シミュレーション結果の可視化を扱っており，可視化対象となるデータは刻々と変化する．したがって，非構造格子を構造格子に変換することなく，直接 Volume Rendering する手法を選択する．

2.4 Cell Projection

本節では，本研究で用いる可視化アルゴリズムである Cell Projection 法のアルゴリズムについて述べる．Cell Projection 法の処理全体の流れを，図 2 に示す．また，以降では格子とセルとを同義的に用いる．

project

まず，与えられた三次元データをスクリーンに投影する．この投影処理は，自身のオブジェクト座標系を持っている元データを，視環境上でのスクリーン座標系に座標変換することを意味する．

scan convert

次に，投影された各セルに対して，scan convert 処理を行う．投影されたセルには，スクリーン上の点 $P(x, y)$ から出る光線に対して，光線が入ってくる面と出て行く面とがある．これらをそれぞれ前面，背面と呼ぶことにする．それぞれの面において，スクリーン上の点 P における関数値 v と，奥行きの座標値 z を，面を構成する頂点から線形補完することによって求める．前面，背面の v, z をそれぞれ $v_{front}, z_{front}, v_{back}, z_{back}$ とすると，P における色値と不透明度は，次の式によって求めることができる．

$$I = c(v_{front}) \otimes c(v_{back}) \times \tau(1 - \exp(-(z_{back} - z_{front}))) \tag{3}$$

$$\alpha = \tau(v_{front}) \otimes \tau(v_{back}) \times \tau(1 - \exp(-(z_{back} - z_{front}))) \tag{4}$$

$\otimes$ は，over 演算子 [9] と呼ばれる，(2) 式における畳み込み演算と同様の計算を行う演算子である．

このようにして求めたセルの P での色値 (RGB)，不透明度 (α) と，z_front, z_back とをひとまとめにしたデータ構造を **ray-segment** と呼ぶ．つまり，scan convert 処理とはセルから ray-segment を計算する処理ということである．scan convert 処理は，画像処理で一般的な scan line アルゴリズムと同様の処理を行う．つまり，隣接するピクセルの計算を行う際に，一つのセルを完全に処理し切るため，メモリのコヒーレンシを活用でき，メモリ効率が良い．

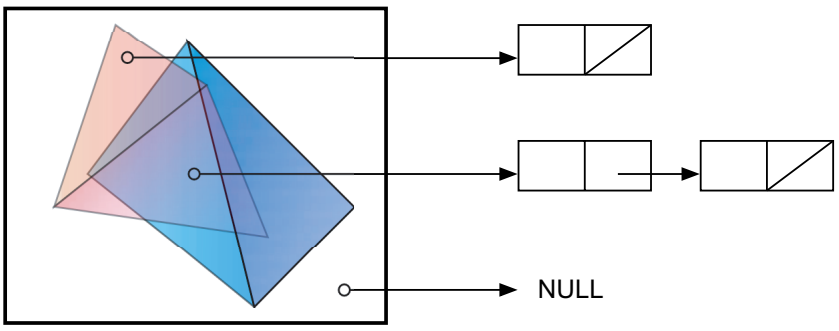


図 3: ray-segment リスト

composite

この scan convert 処理を，各セルに対して行う．結果として，スクリーン

上の各点ごとに，順次 ray-segment が生成され，この ray-segment を奥行き順に並べたリストとして保持しておく (図 3)．このリストにおいて，隣接する ray-segment は奥行き順で $\otimes$ 演算子によって畳み込み合成することが可能であり，これを部分合成と呼ぶ (図 4)．

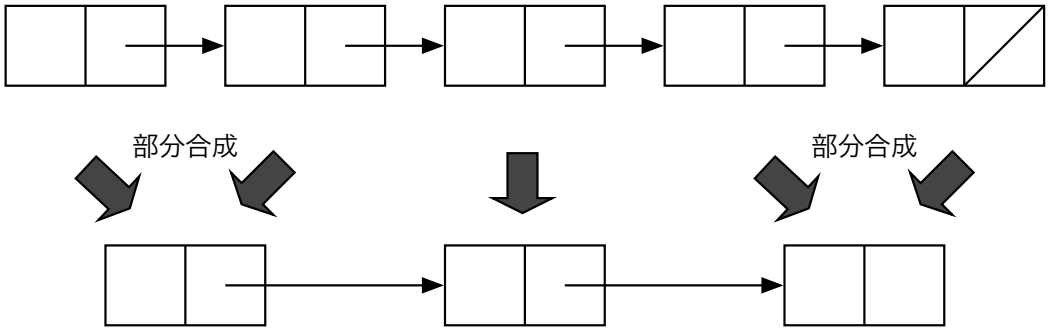


図 4: 部分合成による ray-segment リストの縮小

部分合成した結果もまた ray-segment となり，結果として ray-segment リストを縮小することができる．新しく ray-segment をリストに挿入する際には，既存のリストに対して奥行き順に適切に挿入するため，リストの先頭からたどって挿入を行う．したがって，部分合成によってリストの長さを短くできれば，挿入に要する時間が少なくて済み，高速化につながる．

そして全セルの scan convert 処理が完了した時点で，リストのまだ部分合成されていない全 ray-segment を $\otimes$ 演算子によって完全に合成することで，スクリーン上のピクセル値を得る．

以上のようにして全ピクセルの合成が終わったとき，最終的な結果として三次元数値データを Volume Rendering した画像が得られる．

本研究では，対象とする非構造格子は，単純化のために四面体のみとする．しかし，アルゴリズム自体は他の多面体にも応用可能なものであることに注意されたい．

2.5 Cell Projection の並列化

Cell Projection は，データの配置に関する自由度が高いため，容易に並列化が可能である．以下に，方法を示す．

まず，並列計算機の各プロセッサにデータ (セル) を分配する．各プロセッサ

は，割り当てられたセルを scan convert し，自身の ray-segment リストを構築する．次に，全プロセッサが各自のセルを全て scan convert し終わった後に，全プロセッサの ray-segment リストを，スクリーン上のピクセル毎に一つの ray-segment リストにマージする．その後，マージされたリストを先頭から順に合成していくことにより，リストを一つの ray-segment にする．この過程を最終合成と呼び，最終合成された ray-segment の色値が，そのピクセルでの色を表す．以上の結果，最終的に画像を得る (図 5).

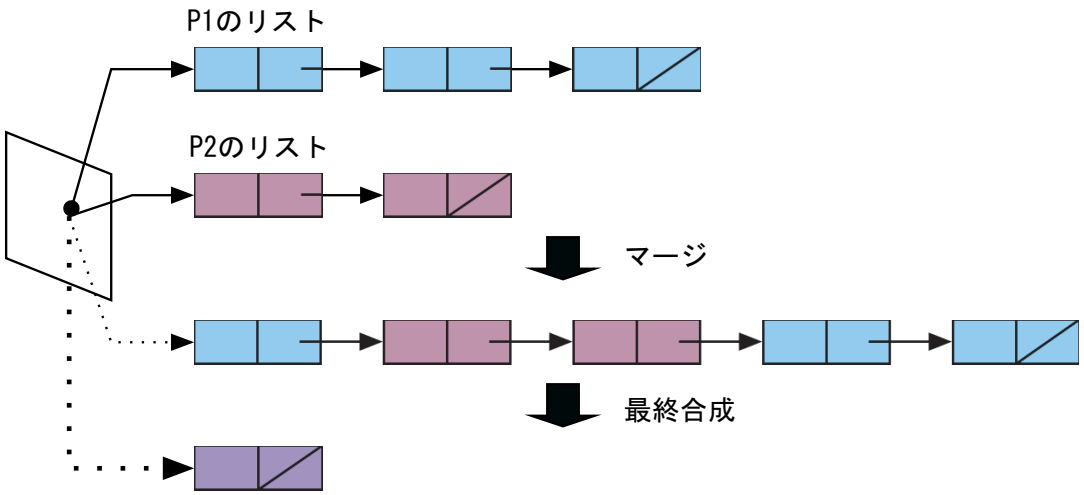


図 5: ray-segment リストのマージ，最終合成

マージする際，プロセッサ間ではスクリーンの解像度に応じた量の ray-segment リストが通信される．したがって，部分合成によってリストを短くできれば，通信量の削減につながり，最終合成の高速化につながる．

2.6 合成

2.4 で述べたように，Cell Projection での主要な処理の一つが，ray-segment の合成である．本章では，部分合成と最終合成について述べる．

2.6.1 部分合成

部分合成が可能なのは，リストに抜けがない場合のみである．すなわち，次の二つのうちのどちらかの場合に限られる．

1. リストで隣接する二つの ray-segment seg_1, seg_2 (奥行き順) において，

$$seg_1.z_{back} = seg_2.z_{front}$$

となり，seg_1 と seg_2 の間に ray-segment が入る余地がない場合

2. seg_1, seg_2 において，

$$seg_1.z_{back} < seg_2.z_{front}$$

であるが，seg_1 と seg_2 の間に ray-segment が存在しないという保証がある場合．すなわち，ドーナツ状のオブジェクトのように，セルとセルの間に空洞が存在する場合

抜けのあるリストは，二つの場合に生成される．

まず，空間的に隣接するセルが異なるプロセッサに分配されている場合である．このとき，あるプロセッサでは抜けのあるリストが生成され，その抜けを埋める ray-segment は，別のプロセッサで生成される．したがって，ray-segment を通信しないことには部分合成ができない．

次に，セルの奥行き順ソートが不正確である場合である．構造格子と異なり，非構造格子では，2.3 で述べたように，セル単位で空間での順序を決めることが不可能な場合がある．スクリーンのピクセル単位においては，セルの奥行き順は一意に決定するが，Cell Projection 法ではセル単位で処理を進める．したがって，セルのソートは不正確にならざるを得ない．このような場合，同プロセッサ内に，抜けを埋める ray-segment に変換されるセルがあるにも関わらず，そのセルの処理が遅れるため，一時的にリストが抜けのある状態になる．

Cell Projection 法において部分合成が可能な場合，最終合成時間の短縮，ray-segment の挿入時のリスト走査時間の短縮など，多くの利点がある．したがって，可能な限り抜けのないリストをつくることが重要である．

2.6.2 最終合成

いま，あるプロセッサが自身の担当するセルをすべて scan convert したとする．このとき，最終合成と同様にそのプロセッサ内で ray-segment リストを全て部分合成することができれば，最終合成に要する通信量はスクリーン解像度に応じた色値だけでよく，ray-segment リストを通信するよりも遥かに通信量を抑えることができる．しかし，2.6.1 で述べたように，抜けのあるリストは部分合成できないので，プロセッサ内で抜けのあるリストが全くない場合を除いては，プロセッサ内合成を行うことはできない．

したがって，最終合成は結局 ray-segment リストを通信する必要があり，リストが長い場合は通信量が大きくなる．そのような大きなリストを一箇所に集め

ると，プロセッサ数が多い場合に通信帯域が不足する．また，一つのプロセッサだけがリストの合成処理を行うということで，負荷の非均衡が発生し，全体の処理時間の低下を招く．

以上のような問題に対し，通信量の削減と合成処理における負荷の均衡をとるため，**Binary Swap Composition Tree (BSC)**[11][12] によって，最終合成を行う．

2.6.3 BSC

N 台のプロセッサで，最終合成を行う場合を考える．BSC は，**交換 (swap)**，**合成 (composite)**，**収集 (gather)** という三つのフェイズから構成される (図 6)．

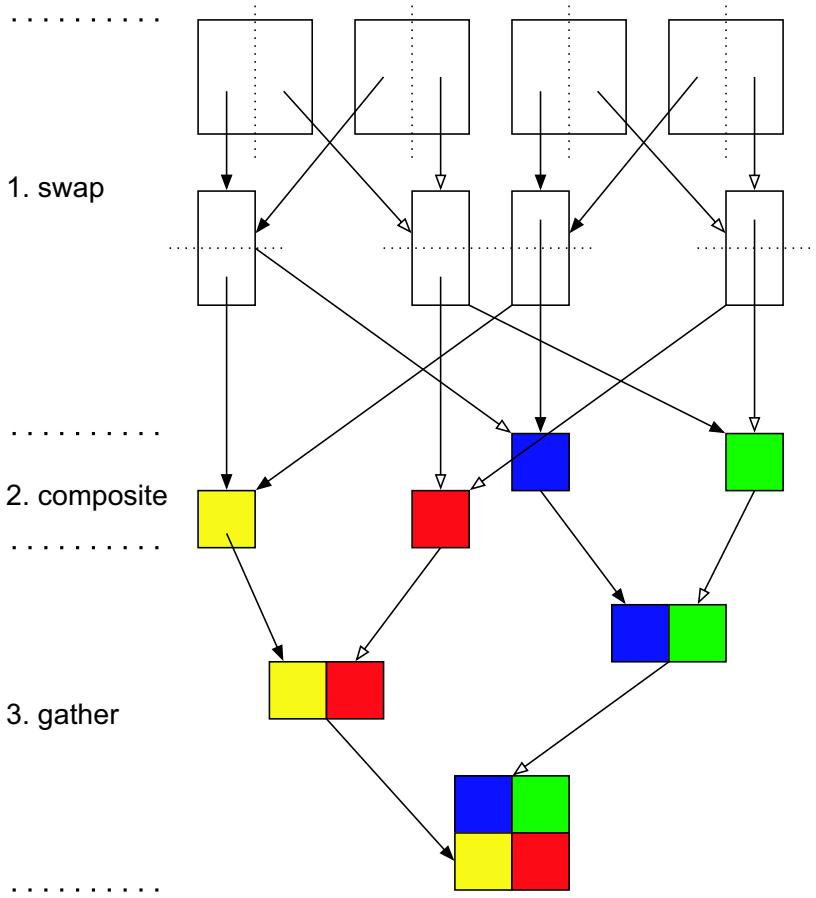


図 6: Binary Swap Composition Tree

swap

1. 各プロセッサは，合成すべきスクリーンを二分割する

2. 自身と同じ領域を持つプロセッサと，互いが異なる領域を担当するようように，分割したスクリーンを交換する

この交換を，$\log_2 N$ 段繰り返す.

composite

$\log_2 N$ 段の交換後には，各プロセッサは自身が担当するスクリーン領域の，すべてのプロセッサによるサブスクリーンを持つことになる．すなわち，最終合成を行うことが可能な条件を満たしている．したがって，各プロセッサが自身の担当領域を最終合成することにより，並列最終合成を行うことができる.

gather

並列最終合成が終わった後では，単純に領域をつなぎ合わせるだけで最終画像を得ることができる．このつなぎ合わせは，N 個の最終合成画像を通常の木による合成を行うことで，一台のプロセッサに集めることで完了する.

BSC 法によって最終合成を行うことの利点は，次の通りである.

通信量

交換のフェイズにおいて，一段交換を行う度に，プロセッサ間でやり取りされるサブスクリーンの大きさは半分になる．単純な一箇所集中合成では，スクリーンの全領域を通信する必要があったのに比較すると，格段に通信量が削減できる.

並列最終合成

単純な一箇所集中合成では，集められてきた巨大な ray-segment リストを，一つのプロセッサがマージし，最終合成を行っていた．それに対し，BSC では，並列に最終合成を行えるため，負荷の均衡につながる．それに加え，各プロセッサが最終合成するサブスクリーンの大きさは，元のスクリーンの大きさの $1/\log_2 N$ で済む.

2.7 Early Ray Termination

Ray Tracing 法や Ray Casting 法では，視点からの光線をスクリーン上の各ピクセルに通過させ，物体に衝突させることによって，そのピクセルでの値を決定する．この光線追跡の際に，十分不透明であるという閾値を定めておき，光線追跡の過程で積算された不透明度が設定した閾値を上回った場合，それ以降の物体からの影響はごく小さいと考え，そこで計算を打ち切る手法が Early Ray

Termination(以下，ERT) である (図 7)．この十分不透明である状態のことを，以降では **terminated** と呼ぶことにする．

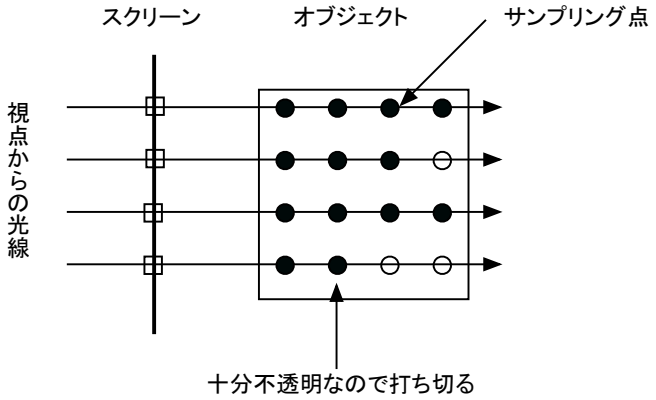


図 7: ERT の概要

ERT による効果は，伝達関数に大きく左右される．すなわち，関数値を高い不透明度に変換する伝達関数を用いる場合は，一般に計算の比較的早期の段階において terminated に達するので，以降の計算を省略できる比率が高いということになる．したがって，表面形状のはっきりしているデータに特に効果を発揮する．Cell Projection への ERT の応用については，第 3 章で詳細に述べる．

2.8 負荷の非均衡

以上で述べてきたように，本研究では，データの配置に自由度がない状況において，非構造格子データを，並列化した Cell Projection 法によって，ERT による最適化を施しながら，分散メモリ型並列計算機で可視化を行う．このとき，以下のような負荷の非均衡が存在する．

形，大きさでの非均衡 Cell Projection 法では，計算時間の多くを scan convert 処理が占め，scan convert 処理の計算量は，投影されたセルの面積に比例する．しかし，非構造格子データの場合，個々のセルの形，大きさが非均質であるため，個々のセルに対する処理量が異なってくる．

視線の向きによる非均衡 各セルの処理量がその投影面積に比例するということは，セルをどのように投影するかによっても処理量が異なるということである．すなわち，ある視環境において投影したセルの面積と，また別の視環境で投影したセルの面積には差があり，非構造格子の場合では，その

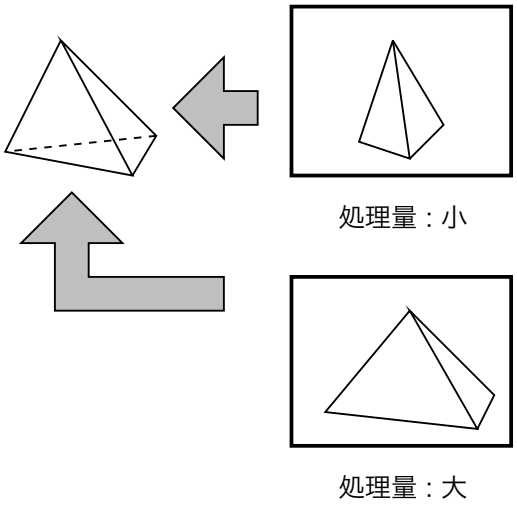


図 8: 投影方向による負荷の非均衡

差が更に顕著なものになる (図 8).

ERT による非均衡　さらに，第 3 章で述べる ERT を行うことによって生じる負荷の非均衡もある．あるプロセッサにおいて scan convert 処理を省略できるセルの量と，他のプロセッサで省略できるセルの量は等しいわけではない．したがって，処理するセルの量がプロセッサ間で異なり，負荷の非均衡につながる．

このように，各プロセッサ間に同数のセルが存在する場合でも，個々のセルの処理量が異なるため，あるいは処理するセルの数が異なるため，プロセッサ間での負荷の非均衡が存在する．

2.9　まとめ

本章では，本研究における背景となる諸技術について述べ，この背景において負荷の非均衡が存在することを述べた．以降，第 3 章では，Cell Projection を高速化するための ERT 手法を提案し，第 4 章では，負荷の非均衡を解決するための動的負荷分散技術について述べる．

第3章　Cell ProjectionへのERTの応用

第2章で述べたように，本研究ではVolume Renderingアルゴリズムとして Cell Projectionを用いている．Cell Projectionでの主な処理は，scan convertと ray-segmentの合成である．ray-segmentの合成は，ピクセル単位で行われる一方で，scan convert処理は投影されたセル単位で行われる．ERTはスクリーン上のピクセル単位で行われる最適化であることを考えると，単純にCell Projectionにおいて ERTを活用できるのはray-segmentの合成処理だけとなる．

ところが，Cell Projection処理に要する時間のうち，大部分を占めるのは，scan convertである．したがって，scan convert処理を省くことができなければ，Cell Projectionにおいては本質的な高速化につながらない．然るに，既存の研究において Cell Projectionに ERTによる最適化を施したという例は未だに見ない．

3.1では，scan convert処理を ERTによって省略するアルゴリズム ERT-table法について述べる．次の3.2では，ERT-table法を並列 Cell Projectionにおいて用いる場合に効率の良い ERTを実現するための，ERT情報の共有手法について述べる．

3.1　ERT-table法

2.3で述べたように，非構造格子のセルを正確にソートすることは必ずしも可能ではない．そこで，本研究における Cell Projectionの実装では，各セルの重心の Z 値によってセルのソートを行っている．もし正確なソートが行えるならば，2.6.1で述べた問題がなくなり，scan convert処理とオーバーラップして最終合成を行うことが可能であり，全体の処理速度の向上が期待できる．

本研究の実装は，scan convertと最終合成をオーバーラップして行わない代わりに，部分合成という次善策を行っている．すなわち，scan convertとオーバーラップして，部分合成が可能な ray-segmentリストは部分合成しようという方針である．いま，図9のような，あるピクセルにおける ray-segmentリストを考える．

ここで，ray-segment s_i, s_j は部分合成可能であり，部分合成を行った結果である ray-segment s_{ij} の不透明度が，十分不透明であるとする．このとき，s_{ij} よりも後ろに生成される ray-segmentは，2.7で述べた ERT法の考え方に従う

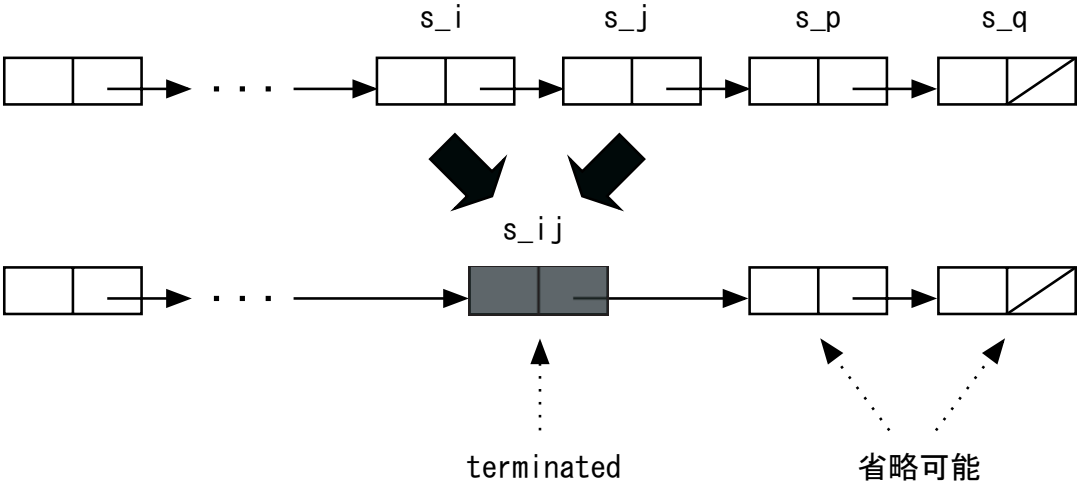


図 9: 部分累積不透明度による ERT

と，処理を省略することが可能である．そして，この省略は s_i よりも前にある ray-segment がまだ求まっていなくとも，あるいは部分合成が不可能である場合においても，可能である．すなわち，あるピクセルでの ERT は，部分累積不透明度の値をもって行うことが可能である．

次に，あるセル c の scan convert 処理を省略できる場合がどのような場合か考える．図 10 のように，c が投影された領域におけるスクリーン上の全ピクセルに対して，それまでに scan convert された，c よりも前方に存在するセルを変換した ray-segment リストが存在し，なおかつそれらのリスト全てが terminated になっているならば，c の投影された領域は既に全て terminated な状態であることになる．したがって，この場合に c の scan convert 処理は省略可能である．[10]

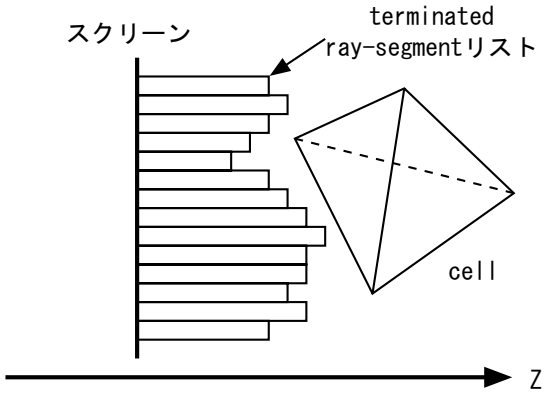


図 10: scan convert を省略できるセル

しかし，このようにピクセル単位で ERT 判断を行うことには，問題がある．各セルを投影した領域にある全ピクセルでの ERT 判断を行うには，scan convert を行うのと同様の反復を繰り返さなくてはならない．ERT 判断を行うには，部分合成済みの ray-segment リストを先頭からたどる必要があるため，この方法では scan convert 処理をしているのと変わらない処理量がかかってしまう．すなわち，ERT のためのオーバーヘッドが大きすぎる．

そこで，厳密にセルの投影面のピクセル単位で ERT 判断を行うのではなく，セルを囲む程度の大きさのサブスクリーン単位で ERT 判断を行う方法を提案する．すなわち，各セル毎に，投影した全ピクセルに対して ERT 判断を行う代わりに，セルを囲むサブスクリーンの全ピクセルが terminated に達しており，なおかつサブスクリーンの全ピクセルのうちで最も Z 値が大きなもの ($deepest$) とセルの頂点のうちで最も Z 値が小さなもの ($nearest$) を比較することにより，ERT 判断を行う (図 11).

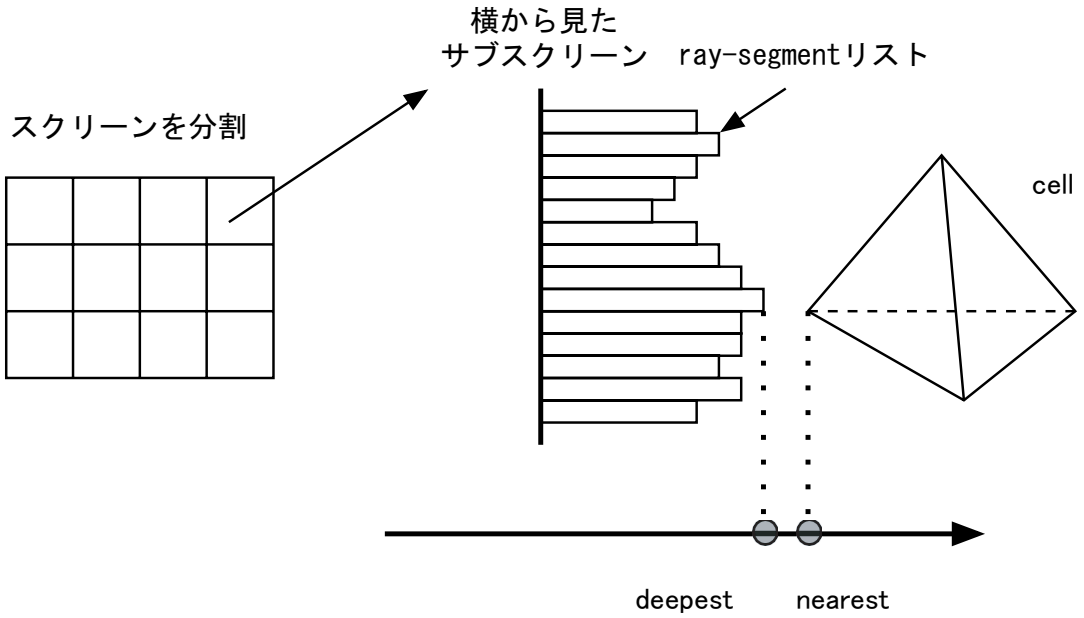


図 11: ERT-table 法

この方法では，あるセルの ERT 判断に要する処理は，セルを囲むサブスクリーンを見つけること，全ピクセルが terminated に達しているかの真理値を調べること，サブスクリーンの Z 値とセルの Z 値と比較，といった単純な処理だけで済み，オーバーヘッドが少ない．一方で，サブスクリーンの全ピクセルが

terminated でないとセルの scan convert を省略できないため，ERT によって省略できるセルの数はピクセル単位で ERT を行う場合と比較して減少する．また，図 10 のように，ray-segment リストの間にセルがはまり込む場合，ピクセル毎では terminated しているため，セルの scan convert は省略できるのだが，$deepest > nearest$ となってしまうため，本手法では scan convert を省略できない．

本手法は，ray-segment の部分合成と密接に関連している．まず，$\otimes$ 演算子による畳み込み演算を行った結果の不透明度は，演算前の ray-segment 同士の不透明度より小さくなることはない．したがって，部分合成を行うことによって，より早く terminate 状態に達することができる．また，ray-segment の不透明度の評価をリストの先頭からたどって行うため，部分合成によってリストを縮小できれば，リストをたどる時間が短縮でき，効率の向上につながる．

3.1.1 データ構造

まず，スクリーンを任意の大きさのサブスクリーンに分割する．このサブスクリーンのことを，**ERT-table** と名付ける．本 ERT アルゴリズムの名称は，この ERT-table に因むものである．

個々の ERT-table は，次のような情報を保持する．

top,left,bottom,right　サブスクリーンの両端点の座標値

pixels　サブスクリーンの各ピクセルに対応する ray-segment リストへのポインタの配列

terminated　*pixels* の各要素が，既に terminate しているか否かを保持する bool 値の配列．全て false で初期化される

count　terminated のうち，true であるものを数える．0 で初期化され，最大値 max_count は，$(right - left) \times (top - bottom)$

deepest　pixels で terminate しているもののうち，最も Z 値が大きい ray-segment リストの Z 値

3.1.2 判定アルゴリズム

投影されたセル c の scan convert 処理を省略できるのは，以下の二つの条件がともに成立する場合である．なお，本手法では投影されたセルが ERT-table に収まらない場合には，ERT を行わない．

1. c の投影領域に対応する ERT-table s を探索する．探索は，c を構成する頂点から，x, y の最大値と最小値をそれぞれ求め，$top_s, left_s, bottom_s, right_s$

と比較することで行う．

ここで，s が発見できない場合，すなわち c の投影領域が複数の ERT-table を横断して存在する場合には，c の scan convert 処理は省略しない．

2. c を構成する頂点から，z の最小値 $nearest_c$ を求める．$nearest_c$ は，c におけるスクリーンに最も近い z 値である．c の scan convert 処理を省略できるのは，s の各ピクセルが全て terminate しており，なおかつどのピクセルの ray-segment リストよりも，c が後ろにある場合である．すなわち，

$$(count = max\_count) \cap (deepest_s < nearest_c)$$

が成り立つならば，c の scan convert 処理は省略できる．

3.1.3 ERT-table の更新

一つの ERT-table の更新処理は，以下のようにして行う．

- *count* が既に最大値の場合，更新しない．
- *pixels* の各要素について，
  - 対応する *terminated* が true ならば，次の pixel へ
  - ray-segment リストの先頭から順に，部分合成結果の累積不透明度を求める
  - その結果得られる ray-segment の近似的な不透明度が閾値を超えているならば，
    - 対応する *terminated* を true に
    - *count* をインクリメント
    - *deepest* と，現在の ray-segment の z_{back} を比較し，大きな方を *deepest* に代入
  - 次の pixel へ

以下に，擬似コードを示す．

```
if(count == (right-left)*(top-bottom)) return;
foreach pixel_i in pixels {
   if(teminated_i == true) continue;
   foreach rs_j in pixel_i {
      if(rs_j.alpha > opaque_enough) {
        terminated_i = true;
        count++;
```

```
        deepest = (deepest > rs_j.z_back) ? deepest : rs_j.z_back;
      }
    }
}
```

3.1.4 考慮すべき諸元

(a) ERT-table の大きさ

本手法では，投影されたセルが ERT-table に収まらないと ERT を行わないため，セルの投影面積よりも大きな面積を持つように，ERT-table の大きさを設定する必要がある．しかし，大きすぎる ERT-table では，count が最大値に収束する，すなわち ERT-table が terminate 状態に収束するまでに時間がかかるようになり，いつまでも ERT が行われないという問題がある．したがって，実際に ERT-table の大きさを決める際には，全セルの投影面積の平均値を少し上回る程度にしておくのが最良であると考えられる．

(b) ERT-table の更新に要する時間

terminate していない ERT-table を更新するには，*pixels* の大きさに比例した時間を要する．したがって，全 ERT-table を更新するのに必要な時間は，スクリーンの解像度に比例する．

実際に，一つの ERT-table の更新に要する時間，そして全 ERT-table の更新に要する時間は，Cell Projection 処理自体を妨げるものであってはならない．最適化のための ERT 処理がオーバーヘッドになっては本末転倒である．

(c) ERT-table の更新頻度

正確な ERT 情報を得るためには，本来ならばセルを一つ scan convert する毎に ERT-table を更新しなければならない．そうすることにより，一つ一つのセルによる ERT-table への影響が反映されるため，全セルに対して ERT 判定を行うことが可能である．しかしその一方で，ERT-table を更新するコストは増大する．

逆に，頻度を下げた場合には，ERT-table を更新するコストは削減されるものの，本来 ERT を適用することのできるセルが，ERT されない可能性がある．

したがって，前項で述べた更新に要する時間を考慮し，できるだけ頻繁に更新するような頻度を定める必要がある．

これらの諸元については，第 5 章で評価を行う．

3.2 ERT 情報の共有

2.1 で述べたように，Volume Rendering はデータ並列性が高い．そのため，大規模なデータをレンダリングするには，並列化が有効な手段となる．そして，前節までで，Volume Rendering において ERT が有効な最適化手法であることを見てきた．したがって，並列 Volume Rendering に ERT を組み合わせるのは自然であり，高速化が期待できる．

しかし，単純に並列 Volume Rendering に ERT を組み合わせただけでは，ERT による効果が完全には発揮できない．例えば，図 12 のように，オブジェクト空間上の異なる領域のデータが，異なるプロセッサ $P1, P2$ に分散して置かれている場合を考える．各プロセッサは，スクリーンに対して $P1$ が手前，$P2$ が奥にあるとする．このとき，視環境によっては，スクリーン方向に領域が重なる．そして，$P1$ のある領域 R_1 において ERT が成立している場合，$P2$ の，R_1 よりもスクリーン方向に対して後ろにある領域 R_2 についても処理を省略することが可能である．ところが，ERT はそれぞれのプロセッサ単位で行われているため，本来省略できる領域について，後ろ側のプロセッサでは処理を行うことになり，無駄が生じてしまう．すなわち，ERT によるセルの scan convert 省略が不十分である．

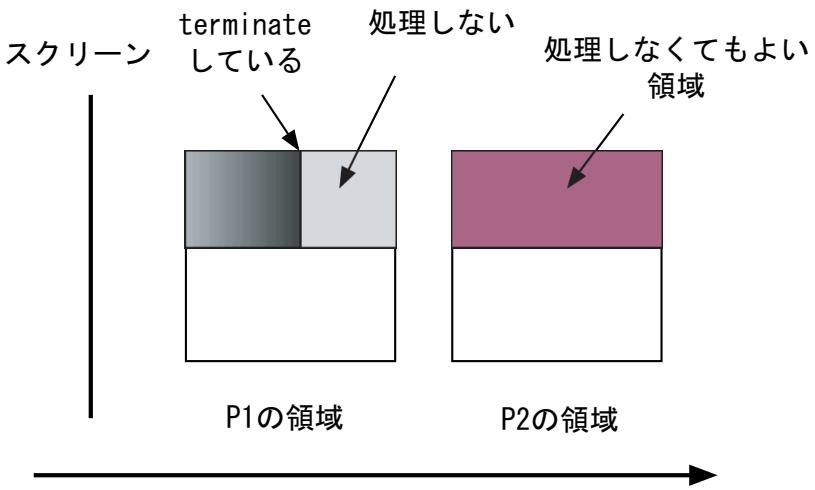


図 12: 不十分な ERT

もし，領域が重なるプロセッサ間で ERT 情報を共有することができれば，全体の処理時間の大幅な削減を期待できる．これを実現するには，一つの管理ノードが全計算ノードの ERT 情報を一元管理し，管理ノードと全計算ノード間で適切な同期をとりながら ERT 情報の更新を行う方法が考えられる．しかし，この方法では，計算ノードの台数が増加した場合に，同期に要する時間が大きくなり，結果としてオーバーヘッドになってしまう．

そこで本研究では，ERT 情報の更新を非同期に行うことによってオーバーヘッドを回避しつつ，ERT 情報を共有することによって全体の処理時間を削減する，Cell Projection における ERT 情報の共有手法を提案する．

3.2.1 アルゴリズム

各計算ノードは，3.1 で述べた ERT-table 法を用いて ERT 処理をローカルに行うため，それぞれの ERT-table をローカルに持つ．したがって，重なる領域を担当する計算ノード間で，重なる領域の ERT-table を共有できればよい．具体的には，管理ノードが重なる領域を持つ計算ノードから，重なる領域の ERT-table を集め，それらの中で deepest が最小であり，なおかつ terminate している ERT-table をもって，その領域での ERT-table とし，領域を担当する計算ノード群に返す．こうすることにより，最も手前で terminate している ERT-table を共有することができる．

実際に通信する ERT 情報は，ERT-table 一つあたり，次のようなものになる．

is-terminated　ERT-table が terminate しているか否かを表す bool 値．すなわち，$count = max\_count$ のときに true

deepest　ERT-table での deepest と同じ

この情報を，必要な領域数だけ管理ノードと計算ノード間で送受信する．

以上のようにして，少ない通信量で ERT 情報の共有を実現することができる．

3.2.2 考慮すべき諸元

共有を行う頻度

頻繁に ERT 情報の共有を行うと，ERT によって省略できるセルの数が増すことを期待できる．しかしその一方で，ERT 情報の共有に要するコスト，すなわち ERT 情報を通信できる形式に変換し，一箇所に集め，その領域における代表 ERT-table を判定し，再び分配し，ローカルの ERT-table に反映させる，といった処理が増大し，scan convert 処理を妨げることになる．

領域数

重なる領域を担当する計算ノード間のみで，重なる領域だけを通信することは理想的である．しかし，重なる領域を管理するコスト，重なる領域だけを特定して通信するのに要するコストを考慮する必要がある．

これらの評価については，第 5 章で評価を行う．

3.3 まとめ

本章では，Cell Projection における ERT 手法として，ERT-table 法を提案した．また，ERT-table 法を並列 Cell Projection において用いる場合における高速化手法として，ERT 情報の共有手法を提案した．第 5 章では，これらの評価を行う．

第4章　動的負荷分散

4.1　負荷分散方式

2.8で述べた負荷の非均衡を解決するには，負荷分散が必要となる．負荷分散方式は，いつ行うかという点で二つに分類できる．一つは，本処理を行う前に全処理空間での負荷量を調べ，各プロセッサが均等な負荷を担当するように分散させる**静的負荷分散**である．もう一つは，本処理を実行する過程の中で負荷の非均衡を検知し，プロセッサ間での負荷を均等にするよう負荷をやり取りする**動的負荷分散**である．

静的負荷分散方式では，負荷分散処理と本処理とが明確に分かれているため，それぞれの処理を集中して行うことができる．しかし，本処理の実行時に負荷状況が変化してしまうような対象には対応できない．

一方，動的負荷分散方式では，負荷分散処理と本処理とが並行して行われるため，互いの処理を干渉してしまう．しかし，本処理の実行時における負荷状況の変化に対応できること，また本処理の開始前に負荷がどれくらいになるか分からない場合においても，負荷の均衡をとることが可能である．

Volume Rendering処理では，視点の変更が頻繁に要求される．このとき，2.8で述べたように，オブジェクトを見る角度によって負荷の状況は変化する．したがって，ある角度では負荷の均衡がとれていたとしても，別の角度では非均衡になると考えられる．Volume Rendering処理を実時間に行うということは，1秒に30回画像を出力するということである．よって，視点が変化するたびに，本処理の前に静的負荷分散を行うと，負荷分散処理がオーバーヘッドになってしまい，実時間性を達成できないと考えられる．

以上の理由により，本研究では動的負荷分散方式を採用する．

4.2　方針

まず，負荷分散の処理単位，すなわち何をもって負荷とするか，何をプロセッサ間でやり取りするか，を決定する．Cell Projectionにおいては，負荷分散の処理単位としてセルのscan convert処理，生成されたray-segmentを合成する処理，の二つが考えられる．しかし，scan convert処理と合成処理のそれぞれに要する時間を比較したところ，scan convert処理の方が多くの時間を要することが分かったため，本研究ではscan convert処理を負荷分散の処理単位とす

る．実際には，セルをプロセッサ間でやり取りすることになる．

本研究における動的負荷分散の基本方針は，以下の通りである．

プロセッサノード構成

負荷分散を管理するための管理ノード (Control Node： **CN**) を一台，残りのプロセッサは並列 Cell Projection を行う計算ノード (Working Node : **WN**) N 台とする．

CNは，最終結果である画像を出力するノードでもあり，また第 1 章で述べたシミュレーションのインタラクションを司る入出力ノードでもある．

負荷移送のタイミング

セルの scan convert が負荷分散の処理単位であるため，計算ノードの残りセルがどれくらいになったときに負荷の移送を行うかという点を考えなければならない．理想的には，scan convert と動的負荷分散をオーバーラップして行うことができれば，WNのアイドル時間を削減することができる．すなわち，残りセルが少なくなってきた WN は，全てのセルを処理し尽くす前にセルの要求を発行することで，セルの受信待ちを回避できる．

ところが，実際には多くのセルがまだ残っている場合でも，ERT の効果によってそれらのセルの大半が処理しないで済む場合，やはり WNはアイドルになってしまう．とはいえ，早すぎるセルの受け渡しは，送信側 WNのセルを早期に減らすことになり，負荷の非均衡の解消が不十分になる．つまり，時間に対するセルの減少率は非連続的であるため，負荷移送のタイミングは一様に決定できない．

以上の理由と実装の単純さを優先させ，本研究では，WNがアイドルになった時点でセルの要求をするという単純な方針を採用する．

受け渡しするセルの量

一回のセルの移送につき，どれだけのセルを受け渡しするか，すなわち負荷分散の粒度をどのように定めるかを決定する．これは，並列処理におけるスケジューリング問題と同等であり，次のような方針が考えられる．

static scheduling　受け渡しするセルの量はあらかじめ一定量に決めておく

guided self-scheduling　最初は多くのセルを移送し，次第に減らしていく

分散メモリ型並列計算機では，通信能力が計算能力に対して圧倒的に不足

している．したがって，通信回数はできるだけ減らし，一度に大きなデータを送受信することが全体の処理速度の向上につながる．つまり，粒度が大きい方がよい．

static scheduling において粒度を大きくすると，処理の終盤において，定められた移送量よりも残りセルの量が下回ること可能性が高くなる．したがって，動的負荷分散が行われる回数が減少し，負荷の非均衡が改善されない．

一方，guided self-scheduling では，残りセルの量に応じた粒度で，移送量が柔軟に決定される．したがって，本研究では，guided self-scheduling を採用する．

具体的には，移送時点における，残りセルの $1/x$ を移送することにし，x については，実測した結果に基づいて決定する．

どの WN から受け取るか

アイドルになった *WN* は，残り *WN* の中のどの *WN* からセルを受け取れば効率がよいのかを元に，送信元 *WN* を選択する必要がある．これについては，次節以降で詳しく検討する．

4.3 送信元 WN の選択

本研究では，動的負荷分散時における，負荷の移送元 *WN* の選択について，以下のものを考える．

4.3.1 random

まだ scan convert していないセルが一定数以上残っている *WN*(busy ノード : WN_{busy} と呼ぶ) の中から，*CN* がランダムに一台を抽出し，セルの送信元とする方針である．以下では，この方針を **random** と呼ぶ．

この方針には，単に WN_{busy} 配列から要素を一つ取り出すだけなので，送信元を決定するコストがほとんどかからないこと，実装が容易であるという利点がある一方，ランダムな選択なので，効率面での保証が全くないという欠点がある．

4.3.2 max-remain

WN_{busy} 配列のうちで，その時点において最も残りセルが多い *WN* をセルの送信元とする方針である．以下では，この方針を，**max-remain** 法と呼ぶ．

この方針は，Cell Projection が基本的にはセル数によって処理量が決定され

ることを考えると，最も直截的なものである．その時点における最大の残りセルを持つ WN の負荷が緩和されることになり，さらに guided self-scheduling を用いているため，その時点での最も大きな粒度でセルが移送されることになり，負荷の均衡につながると考えられる．

一方で，移送元を決定する必要が生じる度に，その時点における残りセルの量が幾つかという小さな通信を WN_{busy} 全体に行う必要があり，WN の台数が増大した場合に CN への通信が殺到してしまうという欠点がある．

4.3.3 bounding-box

ここで，どのような送信元 WN を選べば，受信する WN にとって効率が良いかについて述べる．

セルの scan convert 処理においては，プロセッサが現在持っているセル，それらを scan convert した結果である ray-segment リストに対して，新たにどんなセルを scan convert するにせよ，効率の点で関連はない．したがって，どの WN を選んでも効率に関して差はない．

一方，ray-segment の部分合成に関しては，大きな差が生じる．例えば，受信側 WN において，既にある程度部分合成が行えている領域がある場合に，その領域外にあるセルを受信したとしても，元々部分合成の行えていた領域に対して貢献することはない．部分合成ができない ray-segment リストが増えるだけである．

そこで，重なる領域を持つ WN からセルを受信すれば，受信したセルの中に，抜けのあるリストを埋めるセルが存在する可能性があり，そうした場合，大幅に ray-segment リストを短くすることが可能である．例えば，図 13 において $P1$ と $P2$ が重なる領域 A を持っている場合，その領域間でセルの移送をするとよい．また，抜けを埋めるセルがないにせよ，特定の領域を少数のプロセッサに集中することができるため，部分合成を行える確率が増大し，全体として ray-segment リストの短縮につながる．したがって，最終合成における通信量の削減に貢献し，全体の処理時間を短縮することができる．

また，ERT においても，3.2 で述べたように，重なる領域を担当する WN からセルを受信することにより，ERT による効果が発揮できる確率が向上する．

以上のような理由から，担当するセルのスクリーン上での領域が重なる WN を，送信元 WN に選ぶことによって，効率の良いセルの移送が可能になると考えられる．この考えに基づく方針を，**bounding-box** と呼ぶ．以下では，この

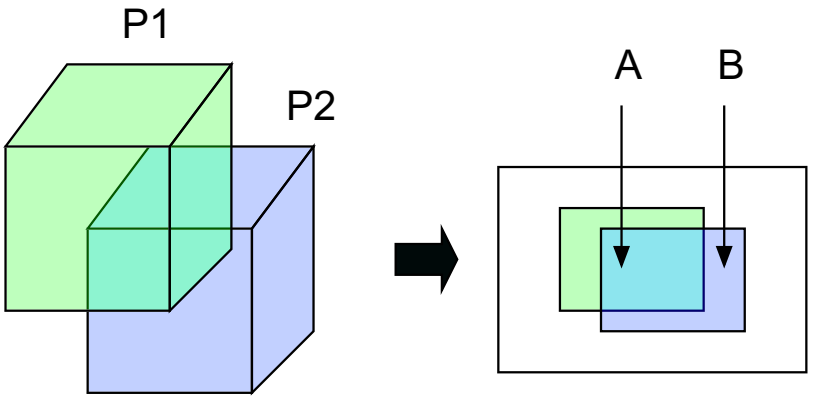


図 13: 重なる領域を持つ WN

方針のアルゴリズムと得失について述べる．

領域の算出

始めに，各 WN が担当するセルの，スクリーン座標系における領域を求める．領域の求め方は，次の二通りがある．

一つは，本研究は，数値シミュレーションと連携した可視化であることを利用して，シミュレーションにおける領域分割情報を活用する方法である．この場合，オブジェクト座標系で与えられる領域情報をスクリーン座標系に座標変換することで，各 WN が持つセルの領域を得ることができる．

もう一つは，可視化側自身で領域情報を計算する方法である．この方法は，シミュレーション側から領域に関する情報が得られない場合や，本研究以外の並列 Cell Projection においては必然的なものである．具体的には，各 WN において，自身の持つセルを構成する頂点全てから，スクリーン座標系における x, y, z 各座標の最大値，最小値を求めることで，その WN が持つ全てのセルを囲む領域である直方体 (**bounding-box**) が得られる．

重なる WN の検出

次に，WN_i において，重なる領域を持つ WN の集合 (dup_i) を求める．

いま，WN_1 と WN_2 との領域 $boundary_1, boundary_2$ における重なりの判定を考える．各 $boundary$ は，それぞれスクリーン座標系における座標値 $top, left, bottom, right$ を保持するとした場合，以下の式を満たす場合は重なりがない．

$$(right_1 < left_2) \cup (left_1 > right_2) \cup (bottom_1 < top_2) \cup (top_1 > bottom_2) \quad (5)$$

レンダリングの本処理に先立ち，各 WN の bounding-box 情報を CN に集め

ておくことにより，重なる WN の検出処理と scan convert とはオーバーラップして行うことが可能である．

セルの移送

WN_i がアイドル状態になったとき，CN は dup_i から WN を一つ抽出し，セルの送信元とする．このとき，どのようにして抽出するかによって効率が変わる．例えば，bounding-box 情報に奥行き方向の最大最小値も含めてあるならば，WN_i よりも手前の領域を持つ WN からセルを移送することで，部分合成や ERT ができる可能性が高まる．あるいは，4.3.2 の max-remain 法と組み合わせることも可能である．

得失

長所については本節の冒頭で述べた通り，重なりのある領域を持つ WN 間でのセルの移送は，部分合成，最終合成，ERT の効率を向上させる点である．以下では，短所について述べる．

まず，本方式は，各 WN の担当するセルが，ある程度空間的に偏っている場合にのみ有効である．例えば，数値シミュレーション側において，静的負荷分散として領域を WN にサイクリック分割している場合や，動的負荷分散を行って領域を分散させている場合を考える．このような場合，空間的に近傍にあるセルは異なる WN に分散して置かれてしまっており，bounding-box がどの WN においても似たものになってしまったり，各 bounding-box が大きくなりすぎるなど，bounding-box としての意味を成さなくなる．

次に，WN がアイドルになった時点で，重なる領域の WN が全セルの処理を終えている場合を考える．このような場合，アイドルになった WN はどの WN を送信元に選べばよいかという問題がある．

したがって，bounding-box 方式は，各 WN にある程度まとまった領域が割り当てられており，なおかつ WN 間に重なる領域が多く存在する場合にのみ，有効な手法である．

また，スクリーン座標系での領域の重なりを調べるため，視環境が変化するたびに bounding-box を再構築する必要がある．

4.3.4 OcTree

4.3.3 の bounding-box 方式は単純ではあるが，重なる領域を持つ WN 間で単純にセルの移送を行うだけでは，重ならない領域に属するセルを移送してしまう可能性があるという点において，不完全でもある．例えば，図 13 において，$P1$ が

アイドルになったときに，$P2$ が B の領域からセルを渡してしまうと，領域間の重なりがない．このような場合，bounding-box 方式の利点を生かすことができず，部分合成や ERT による効率の向上が望めない．また，抜けのあるリストを埋めることができるならば，リストを大幅に縮小できる．しかし，bounding-box 方式では，スクリーン座標系における特定の領域に含まれるセルの移送が保証されているのみであり，より具体的な位置を特定したセルの移送を行うことができない．

これらの問題は，セルの位置をより細かく指定して移送することができれば解決できる．そこで，本節では bounding-box 方式を改善する **OcTree** 方式について述べる．

OcTree (八分木) とは，空間を再帰的に八分割 (x, y, z 方向) することによってオブジェクトの形状を表現するデータ構造である．空間を占めるデータの量に応じて適応的に分割を繰り返していくことにより，効率的に形状を表現することが可能である．

本方式では，この OcTree を用いることによってセルのオブジェクト座標系における位置管理を行う．すなわち，本研究における並列 Cell Projection では，基本的にセルのストレージとしてソート済みの一次元配列を用いているのに対し，本方式は，OcTree をセルのストレージとする．

データ構造

各 WN で，始めに空の OcTree (*root*) を用意しておく．そして，自身の担当する全てのセルを，OcTree に格納していく．本研究で用いる OcTree のデータ構造は次の通りである．

cells　格納しているセルへのポインタ配列

children　子の部分木へのポインタ配列．OcTree なので，8 要素から成る配列

spawn_enough　自身が持つセルの量の最大値．*cells* の量がこの値を超えると分割を行い，*children* にセルを引き継ぐ

max_descent　分割を行う最大値

アルゴリズム

ある部分木 T におけるセル c の挿入は，次のようにして行われる．

- T が既に子を持っているならば，
  - c を囲む bounding-box b_c を求める
  - b_c を内包する T の子 t を *children* から探す．

もし，内包する子がいない，すなわち c が複数の子を横断する場合は，T の $cells$ に c を挿入し，終了する

  - t に対して，再帰的に c の挿入を行う
- T に子がいない場合は，自身の $cells$ に c を追加する

各 WN は，自身が持っているセル全てを $root$ に対して挿入することにより，OcTree の構築を行う．この構築は，Cell Projection の前処理として行う．

得失

本方式では，OcTree をセルのストレージとして用いることにより，特定の領域にあるセルを容易に指すことが可能になる．また，OcTree の最大分割数である $max\_descent$ を大きくすることで，より細かな領域指定が可能になる．さらに，セルの移送時には部分木ごと移送することによって，特定の領域にあるセルを一括して移送することが可能である．

したがって，重なりのある WN 間で，真に重なっている領域に属するセルだけを移送することが可能である．例えば，図 14 において，$P1$ と $P2$ の重なる領域 D だけを指定することができる．また，WN_i で抜けのあるリストが存在するとき，抜けを埋めるセル c を持つ WN_j を見つけることができ，さらに c を的確に WN_i に移送することも可能である．

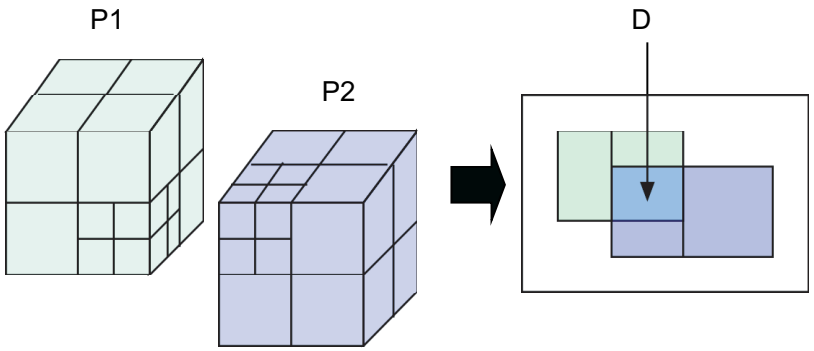


図 14: OcTree での領域指定

また，OcTree はオブジェクト座標系で構築するため，視環境が変化しても OcTree 構造を更新する必要はない．この点で，OcTree 方式は bounding-box 方式よりも優位である．

一方，もし前処理として OcTree を構築するのに要する時間が大きければ，本方式でのオーバーヘッドとなる．OcTree を再構築するのはデータが変更された

ときのみであり，視環境が変更されたときには再構築しないでよいことを考えると，本方式は，頻繁に更新されるデータを生成する系においては適さないが，データの更新頻度が低く，様々な視環境で可視化を行うような場合には適しているといえる．

4.4 まとめ

本章では，負荷分散として動的負荷分散を用いること，セルを負荷移送の単位とすること，四つの方針の検証を行った．次章では，これらの方針の評価を行う．

第5章　評価

以上で述べてきたERT，BSP Composition Tree，各動的負荷分散の効果について，またそれらを総合した場合の評価を行う．評価環境は，表1の通りである．

表1: 評価環境

台数	CPU	主記憶	ネットワーク	OS	並列ライブラリ
8	Pentium4 3GHz	1GB	1000BASE-T	Linux 2.4.21	LAM-MPI 6.5.9

評価に用いた三次元データは，正常者の動脈弓に対してセグメンテーションを行なった後の, inp形式のUCD(Unstructured Cell Data)データ[1)]であり，各セルの形状は四面体の非構造格子である．セル数は307565,，頂点数が62475であり，各頂点はそれぞれ圧力値$P(N/m^2)$，速度ベクトル$v(m/s)$を浮動小数点型の数値データとして持つ．このデータを可視化した例を，図15に示す．本研究

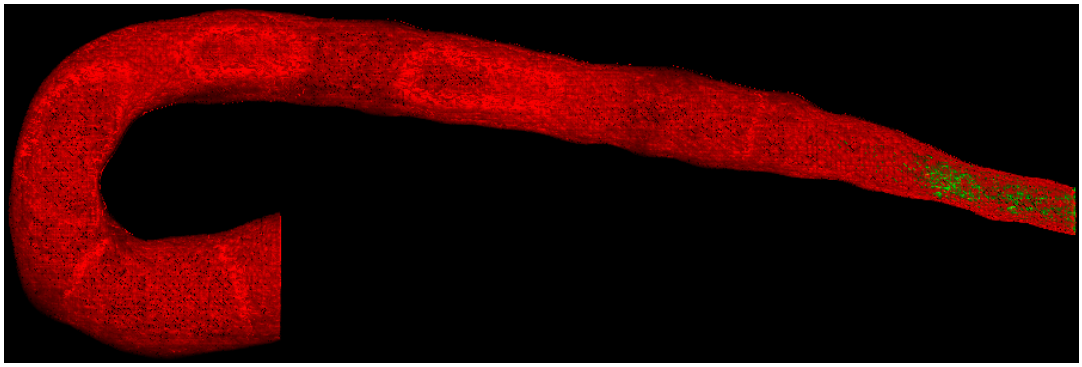


図15: 動脈弓データ

では，ERTによる効果を見るため，伝達関数は不透明度を高めに設定する．具体的には，関数値を変換後の不透明度の平均値が0.265になるように設定した．

このデータでは，全てのセルと頂点が一つのファイルとして集められているため，本研究の目的であるシミュレーションと連携した可視化を模倣するよう，各WNにあらかじめ分配しておく必要がある．4.3.3で述べたように，セルの分

[1)] inp形式のUCDデータは，汎用可視化ソフトウェアAVSで用いられているファイル形式である．

散配置は本来，数値シミュレーション側に依存するが，実際のシミュレーションにおいては，近傍のセルは同じプロセッサに置くことが多いと考えられる．したがって本研究では，以下のようにして空間的に近傍にあるセルを集め，各 WN に分配する方式をとる．

1. 全セルから任意のセル c_0 を抽出する
2. c_0 を構成する四頂点のうち，三頂点を含むセルを探索することで，c_0 に隣接するセルの集合 $neibor_0$ を見つける．
   このとき，$neibor_0$ の要素のうち，既に見つかっているセルは除外する
3. 各 $neibor_0$ に対して，2 と同様にして隣接セルを探索する
4. 以降，幅優先探索と同様の探索を続けることで，近傍のセルの集合を得ることができる

以上の処理を WN の台数分だけ行うことにより，各 WN に割り当てるセルの集合を得る．このとき，各 WN に同数のセルを割り当てる．各 WN に割り当てられているセルを，Cell Projection 法によって可視化したものを，図 16 に示す．

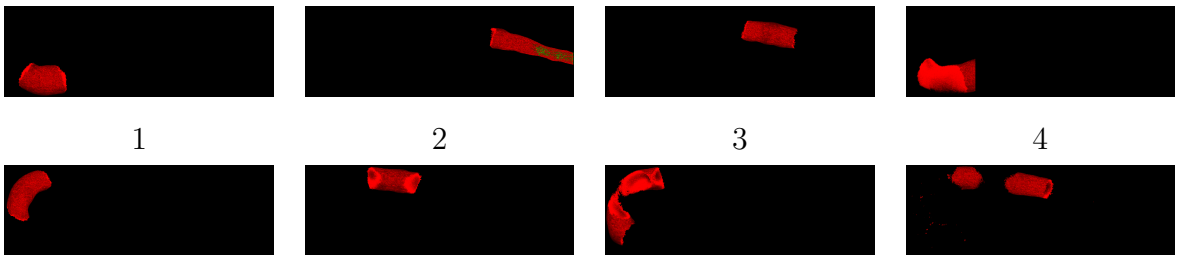


図 16: 各 WN が持つセル

次に，評価を行う視環境について述べる．空間的に偏ったセルの分配を行ったため，視野角によってセル領域の重なりが多い場合と少ない場合が生じる．これまで述べてきたように，重なりの度合いは，ERT や部分合成に大きな影響を与える．したがって，scan convert に関する評価に関しては，(A) 重なりの少ない視野角，(B) 重なりの多い視野角 の二通りについて行う (図 17)．図の短形は，各 WN が持つ全セルの bounding-box を表したものである．図 16 は，(A) の視環境における可視化の例である．

(A) (B)

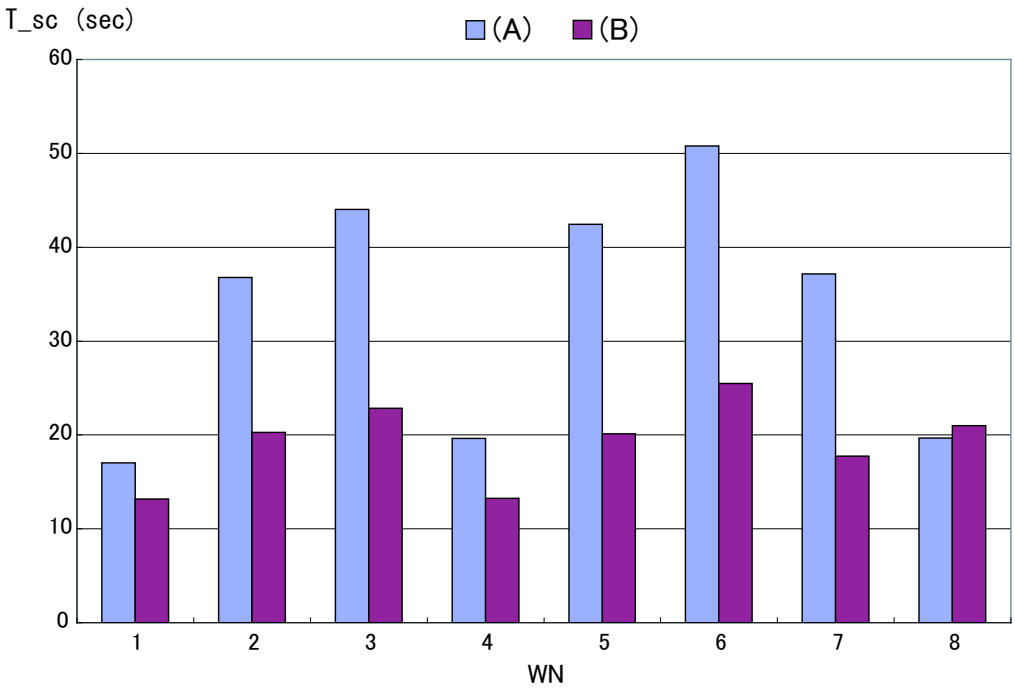
図 17: 視環境

5.1 負荷の非均衡

始めに予備評価として，前述のように分散配置したデータを，並列 Cell Projection によって可視化を行うときの負荷の非均衡について調べる．

評価は，各 WN が自身の担当するセル全てを scan convert し終えるまでの時間 T_{sc} を計測する．視環境は，(A)，(B) の両方を用い，それぞれのスクリーン解像度は，(A) が 900×300，(B) が 300×300 とする．結果を図 18 に示す．横軸は WN のノード番号，縦軸は $T_{sc}(sec)$ である．

図 18: 各 WN での scan convert 時間 $T_{sc}(sec)$



いずれの視環境においても，同数のセルを処理しているのにも関わらず，全

セルを scan convert する時間に大きな差がある．すなわち負荷の非均衡が生じていることが分かる．

5.2 ERT

まず，ERT による，scan convert 処理の省略の効果について検証する．本評価全体における条件は，次の通りである．

- terminate する閾値を，0.9 と定める
- ERT のみの評価を行うため，一台のプロセッサにおいて全セルを処理する
- それぞれのスクリーン解像度は，(A) が 300×100，(B) が 300×300

5.2.1 ERT-table の大きさ，更新のコスト

ERT-table の大きさは，スクリーンの分割数によって決定される．したがって，適切な ERT-table の大きさを決定するため，スクリーンの一辺の分割数 D を変化させたときの，

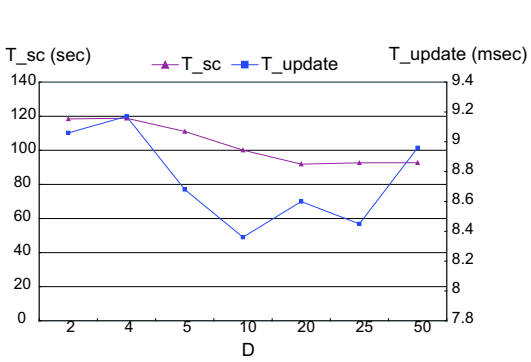
- 全セルを scan convert し終えるまでの時間 T_{sc}
- 全 ERT-table を一度更新するのに要する時間 T_{update}
- ERT によって scan convert を省略できたセル数 N

について，評価を行う．評価条件は，次の通りである．

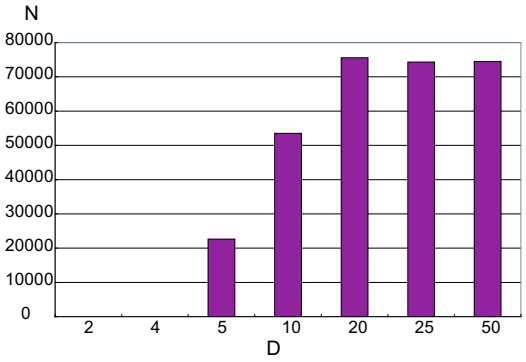
- ERT-table の更新は，全セル数の 1/10 間隔で行う
- 部分合成は行わない
- 視環境は，ERT の効果が顕著である (B) で行う

結果を図 19(1)，(2) に示す．(1)，(2) ともに横軸は D であり，(1) の左縦軸は $T_{sc}(sec)$ を，右縦軸は $T_{update}(msec)$ であり，(2) の縦軸は N (個) である．

このデータ及び視環境では，$D = 20$ のとき，すなわち幅と高さが 15 である ERT-table のときに，T_{sc} が最も小さく，かつ N が最も大きくなり，最良である一方で，T_{update} は三番目に小さな値であり，バランスがとれている．このことから，いたずらに分割数を上げても，セルが ERT-table の中に収まりきらないために ERT の効果は出ず，むしろ ERT-table の更新に要する時間が scan convert の妨げとなっていることが分かる．図 20 は，投影されたセルの x, y 方向の最大の長さの分布図である．この図から，多くのセルを投影した大きさが 15×15 の範囲に収まっていることが分かり，$D = 20$ で最適であることと一致する．したがって，与えられたデータに対し，可視化の前処理としてセルの大きさの平均値を調べておき，その平均値よりも少し大きい値に D を設定するのがよいと言



(1) T_{sc}, T_{update}



(2) N

図 19: ERT-table の大きさ，更新のコスト

える．

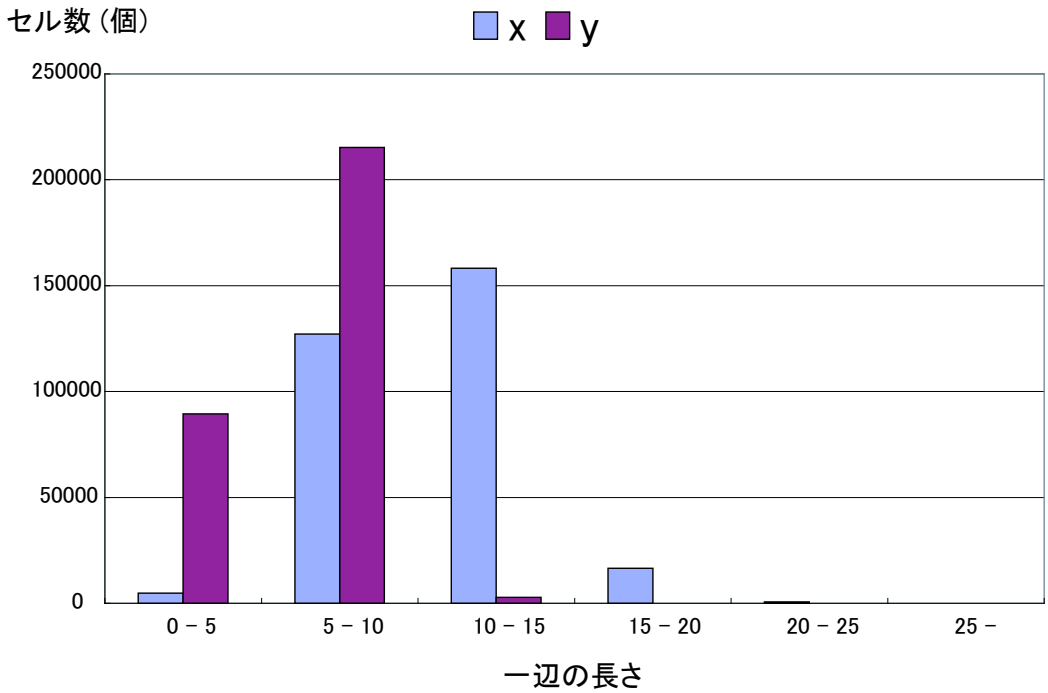


図 20: セルの大きさの分布

さらに重要なのは，ERT-table の更新に要する時間が，全 scan convert 時間の 0.1% にも満たない点である．したがって，本手法では，適切な ERT-table の大きさを選択することにより，scan convert の妨げにならずに，性能向上を実現することが可能であるといえる．

5.2.2 ERT-table の更新頻度

本研究では，セルの scan convert を何回か行う度に，全 ERT-table を更新する方法をとる．そこで，いくつセルを scan convert する度に，ERT-table 全体を更新するかを検証する．ERT-table 更新間隔を，(その時点における全セル数)/定数 C とし，C を変化させた場合における，

- 全セルを scan convert し終えるまでの時間 T_{sc}
- ERT によって省略できたセルの scan convert 数 N

について行う．条件については前項と同様に，ERT-table の大きさは，前項の結果より $D = 20$ とする．結果を，表 2 に示す．

表 2: ERT-table の更新頻度

C	10	100	1000	10000	100000
T_{sc} (sec)	91.93	118.5	226.96	578.749	575.325
N (個)	135352	104415	35731	25294	25294

予想に反し，短い間隔で更新する方が N が少ない．すなわち，scan convert を省略できたセル数が少ない．この理由として，二点考えられる．まず，頻繁に ERT-table を更新することによって，ray-segment リストをたどる回数が増える点である．すなわち，scan convert が進むにつれ，リストの長さは単調増加するため，頻繁に ERT-table を更新することの利点がなくなってしまっている．したがって，部分合成を行うことにより，改善が期待できる．もう一つは，本来ならば ray-segment リストの前の方で terminated に達するにも関わらず，頻繁に ERT-table を更新することにより，*deepest* が過剰に大きくなってしまう点である．これについても，部分合成を行うことによる改善が見込める．また，更新によって *deepest* が大きくなる場合，その更新を破棄することも考えられる．

5.2.3 部分合成による ERT への影響

前項を踏まえ，2.6.1 で述べた，部分合成による ERT の収束の早期化について評価を行う．評価対象，条件は前項と同じである．

ERT-table の更新に先立って，部分合成を行う場合の結果を表 3 に示す．

T_{sc} に関して，C が 1000 までの範囲で，部分合成しない場合に対して 2 倍程度の性能向上が見られる．また，N に関しても，C に対する減少が緩やかに

表 3: 部分合成による ERT への影響

C	10	100	1000	10000	100000
T_{sc} (sec)	50.93	43.99	94.59	540.56	523.21
N (個)	206506	217851	201631	197974	192013

なっており，部分合成による ray-segment リストの縮小の効果が出ている．一方，$C > 1000$ の範囲では T_{sc} が部分合成をしない場合と同程度になっており，実用的ではない．

以上述べてきた，部分合成の有無についての比較を，図 21 にまとめて示す．横軸は $\log_{10} C$ であり，縦軸は N, T_{sc} である．N については，値が大きいほど

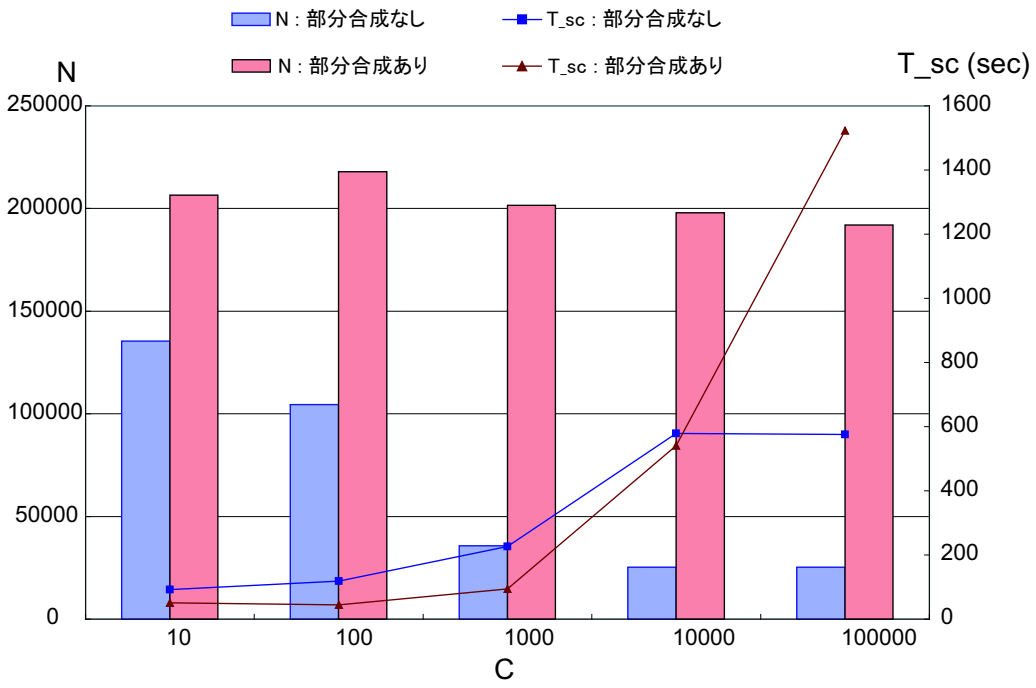


図 21: ERT における部分合成の有無

scan conevrt を省略できたセルが多いことを示す．逆に，T_{sc} については，値が小さいほど，scan convert に要した時間が少なくて済むことを示している．このように，$C = 100$ でどちらの値に関しても最良であることが分かる．

5.2.4 ERT による scan convert 処理の削減

以上の結果を基に，ERT を行う場合と行わない場合の比較を行う．

視環境は図 16 に示した (A)，(B) の両方を対象とする．部分合成を行い．ERT を行う場合，前項までで得られた知見から，$D = 20$，$C = 100$ とする．評価は，全セルの scan convert 時間 T_{sc} について行った．結果を表 4 に示す．

表 4: ERT による scan convert 処理の削減

ERT の有無 / 視環境	無し	有り
(A)	49.34	39.06
(B)	169.87	43.99

領域の重なりの少ない (A) では，ERT による効果があまり出ない一方で，領域の重なりの多い (B) においては，3.86 の高速化を達成した．

5.3 ERT 情報の共有

次に，ERT 情報の共有による，scan convert 処理の省略の効果について検証する．

本研究では，次のようにして ERT 情報の共有を実装した．

- 並列 Cell Projection を行っている各 *WN* は，自身が scan convert したセルを *count* 変数で数えておく
- *count* が一定量増加する毎に，scan convert を中断し，自身の持つ全 ERT-table を *CN* に送信する
- *CN* は，全 *WN* における ERT-table の集合 T を管理する
- *CN* は，送られてきた ERT-table の集合 t と，その時点における T とをマージする
- *CN* は，マージした ERT-table の集合を，全 *WN* にブロードキャスト送信する

以上の結果，ある *WN* での ERT-table を，定めた間隔で全 *WN* に共有することが可能になる．

本評価における条件は，次の通りである．

- terminated に達したとみなす閾値を，0.9 と定める
- 8 台による並列 Cell Projection

- ERT 情報の共有のみの評価を行うため，並列 Cell Projection において負荷分散を行わない
- それぞれのスクリーン解像度は，(A) が 900×300，(B) が 300×300

まず，ERT 情報の共有を一度行うのに，どれだけの時間を要するかを調べる．すなわち，次の処理に要する時間 T_{share} を計測する．

1. WN_i の全 ERT-table の構造体を，MPI で送信可能なデータ型に変換
2. WN_i から CN へ，変換済み ERT-table データを送信
3. CN は，受信したデータを基に T を更新
4. CN は，全 WN に更新済みの ERT-table データをブロードキャスト送信
5. 各 WN は，受信した ERT-table データを基に自身の ERT-table を更新

評価条件は，次の通りである．

- 部分合成，ERT，ERT 情報の共有を行う
- 8 台での並列 Cell Projection
- 視環境は (A)，(B) 両方について行い，それぞれのスクリーン解像度は $900 \times 300, 300 \times 300$ とする

評価の結果，(A) では $T_{share} = 2.5msec$，(B) では $T_{share} = 0.6msec$ となり，非常に高速に ERT 情報の共有が行えている．したがって，T_{share} は全処理時間に対して十分に小さく，ERT 情報の共有には，ほぼオーバーヘッドがないといえる．

本研究では，重なる領域を持つ WN 間のみで ERT 情報を共有する方針はとらず，全ての WN で共有する方法を採用している．この場合，領域数 D は，ERT-table の大きさで決定される．すなわち，

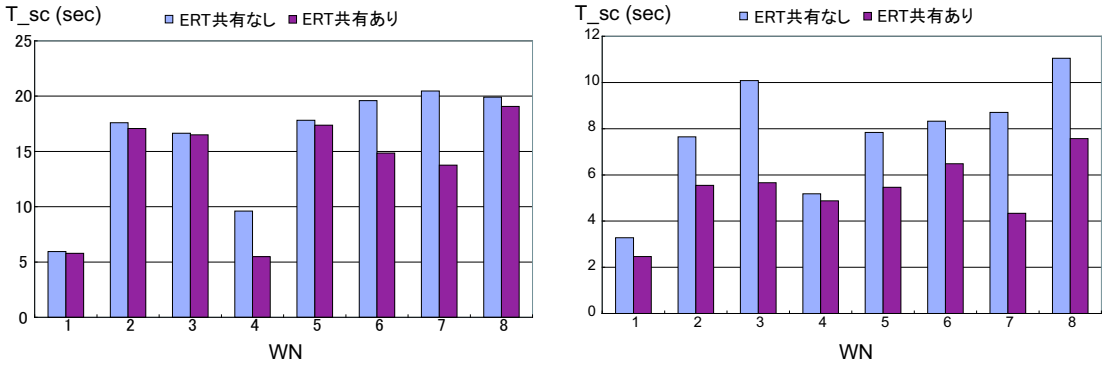
$$D = (width_{screen}/width_{ert-table}) \times (height_{screen}/height_{ert-table})$$

したがって，この評価は 5.2.1 の結果に従う．

5.3.1 ERT 情報の共有による scan convert 処理の削減

以上の結果を基に，ERT 情報の共有を行う場合と行わない場合の比較を行った．結果を，図 22 に示す．横軸は WN のノード番号，縦軸は $T_{sc}(sec)$ である．

重なりの少ない (A) においては，わずかに重なっている $WN = 1, 4$，$WN = 6.7$ で，ERT 情報の共有による高速化が見られる．それに比べて，重なりの多い (B) においては，全 WN において大幅な性能向上が見られる．中でも，$WN = 7$ に関しては，1.99 倍の高速化に成功している．



(A) (B)

図 22: ERT 情報の共有による scan convert 処理の削減

以上の結果より，ERT 情報の共有手法は，セル領域の重なりが多い場合に非常に有効であることが分かった．今後，台数を増やした場合，さらにに重なる領域が増えることを考えると，ERT 情報の共有手法は，並列 Cell Projection において非常に有効な手法といえる．

5.4 BSC

次に，単純な一箇所集中合成と，BSC での合成との比較を行う．評価は，全 WN における scan convert が終了した時点から，CN で最終合成が完了するまでに要する時間 T_{cp} を計測する．評価条件は，次の通りである．

- 動的負荷分散，ERT，ER 情報の共有は行わない
- 8 台による並列 Cell Projection
- 最終合成における通信量，通信時間を見るため，(A)，(B) の両方について，また部分合成を行わない場合 X，行う場合 Y について評価を行う．それぞれのスクリーン解像度は，(A) が 900×300，(B) が 300×300 とする

評価結果を，図 23 に示す．

部分合成を行わない X では，ray-segment リストが長いため，最終合成に要する通信量が多い．したがって，BSC による効果が大きく，(A)，(B) 双方において約 4 倍の性能向上を達成している．また，部分合成を行っている Y においても，約 3 倍の性能向上が見られる．

また，実装の容易さに関しての評価として，それぞれの場合におけるプログラムの行数を数える．実装は C++を用いて行っており，単純な一箇所集中合成

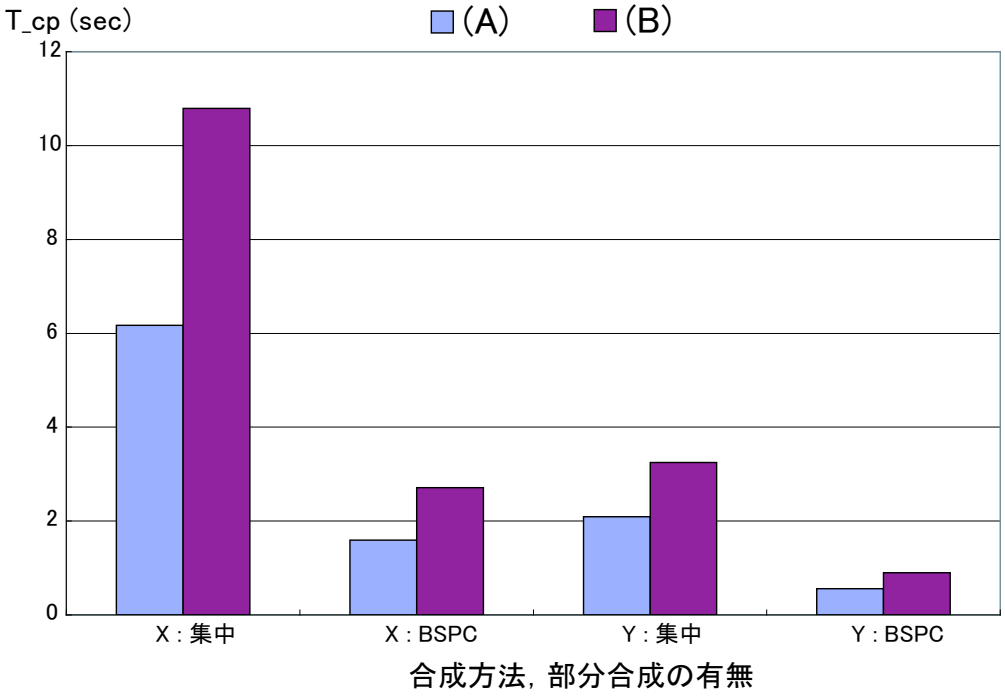


図 23: 合成方法の比較 T_{cp}

では，49 行であるのに対し，BSC では 130 行であった．したがって，BSC 法は，容易に実装でき，なおかつ数倍の性能向上を達成できる．

5.5 動的負荷分散

本節では，4.3 で述べた送信元 *WN* の選択方法について，評価を行う．まず，OcTree 方式の評価を行った後に，他の選択方法について評価を行う．

5.5.1 OcTree

始めに，再帰的分割を行う最大の深さ *max_descent* 及び OcTree に挿入するセルの数を変化させた場合における，OcTree の構築時間の評価を行った．評価は，一台のプロセッサで行った．また，OcTree の構築はオブジェクト空間で行われるため，視環境は無関係である．結果を図 5.5.1 に示す．

この結果より，OcTree の構築に要する時間は，*WN* の持つ全セルを scan convert する時間に比べて十分に小さく，scan convert 処理のオーバーヘッドにはならないことが分かる．

次に，OcTree をセルのストレージとした場合における，各 *WN* での scan con-

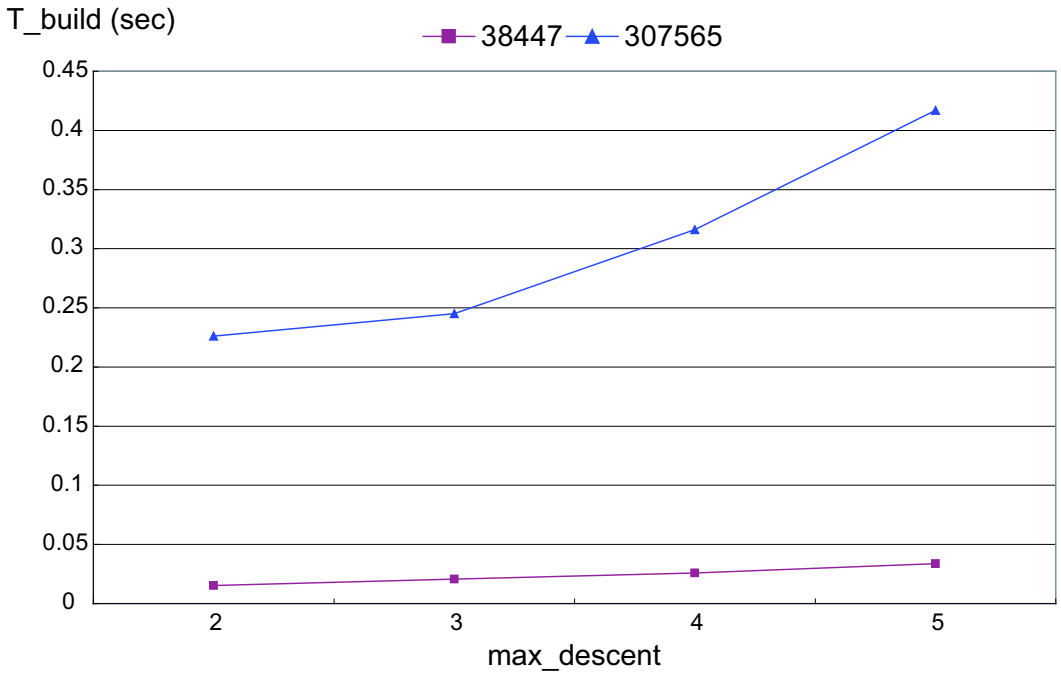


図 24: OcTree 構築時間 T_{build}

vert 時間を計測する．各 WN では，その時点での視環境において手前側にある部分木から順に，再帰的に OcTree に含まれるセルを scan convert する．こうすることにより，視環境が変化する度にセルのソートをやり直す必要が無くなる．ある視環境における部分木の走査順は，図 25 のとおりである．この順序を求めるには，次のようにする．

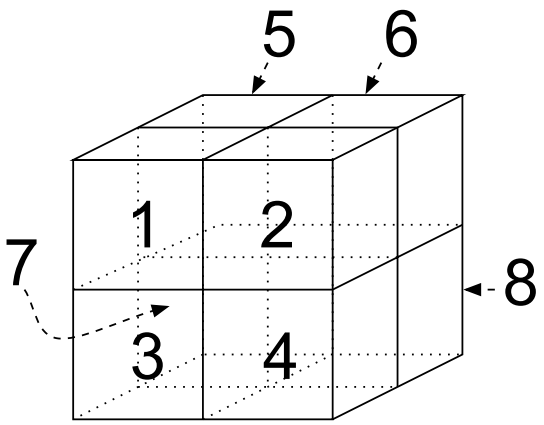


図 25: 部分木の走査順

1. オブジェクト座標系での Z 方向ベクトル v_o と，視環境に置ける視線の方向ベクトル v_e との角度 rad を求める
2. rad が，$45 + (90 \times n)$ を超えると，部分木の走査順が変化する
3. この走査順は，$360/90 = 4$ 通りであるため，走査順を 8 要素から成る静的配列として 4 通り用意しておき，rad によってそれらを参照する

以上の方法によって，OcTree での scan convert 時間 T_{sc} について評価を行う．評価は，WN_1 を用い，視環境は (B) とし，スクリーン解像度は 300×300 である．ストレージを通常の配列とした場合と，OcTree とした場合について，$max\_descent$ を変化させて比較を行い，結果を表 5 に示す．

表 5: OcTree での scan convert 時間 (sec)

$max\_descent$	2	3	4	5	通常の配列
T_{sc}	8.61	12.19	30.72	130.51	3.31

通常の配列の場合と比べ，OcTree の方法において大幅に性能低下が生じている．$max\_descent$ が小さな場合における原因としては，空間分割が大きいため，部分木に含まれるセルの順序が著しくばらばらであり，セルをソートしていないことによって部分合成と ERT を行う率が低下することが考えられる．$max\_descent$ が大きな場合における原因としては，部分合成と ERT を行う率は向上する一方で，OcTree を走査する時間によるオーバーヘッドによって性能が低下していると考えられる．従って，$max\_descent$ を大きくすることなく，各部分木でセルをソートする方法を試み，結果を表 6 に示す．

表 6: 各部分木でソートする OcTree での scan convert 時間 (sec)

$max\_descent$	2	3	4	5	通常の配列
T_{sc}	5.35	9.57	29.60	135.21	3.31

ところが，各 $max\_descent$ の値における改善は微小であり，$max\_descent = 5$ においてはむしろソートしない場合よりも性能が低下している．この原因は，各部分木でのセルのコピー，ソートに要する時間がオーバーヘッドになっているからであると考えられる．

以上の結果から，通常の配列方式に比べてオーバーヘッドが大きすぎるため，本研究では動的負荷分散方式としては採用しない．

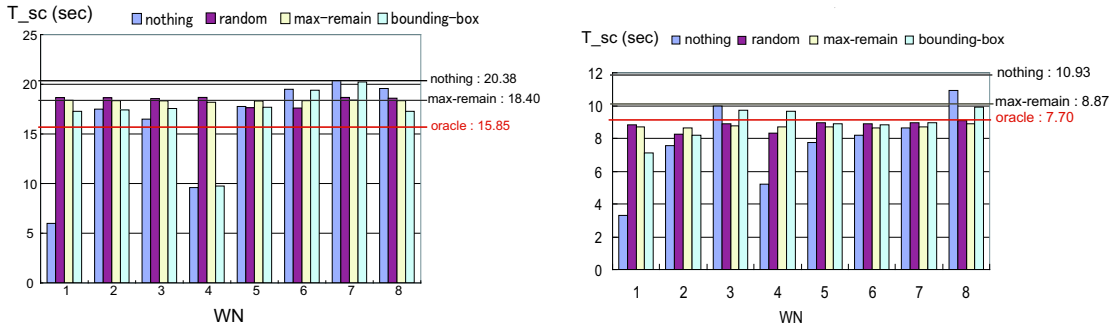
OcTree を構築してもセルはソートしておくことが，実用的な性能を発揮するために必要である．したがって，OcTree 方式を改善する方法としては，視環境が変化する度に各 WN は自身の持つ全セルをソートし，ソート済みのセルの配列を保持し，OcTree ではその配列の要素へのポインタによって，セルを管理するといった方法が考えられる．

5.5.2 各方式の比較

5.5.1 で棄却した OcTree 方式以外の動的負荷分散方式について，評価を行う．本評価における条件は，次の通りである．

- 8 台による並列 Cell Projection
- 部分合成，ERT を行う．各方式の比較のみを行うため，ERT 情報の共有は行わない
- 視環境は (A)，(B) 両方について行い，それぞれのスクリーン解像度は $900 \times 300, 300 \times 300$

評価は，各 WN での scan convert 時間 T_{sc}，全処理時間 T_{all} を計測し，各視環境での結果を，それぞれ図 26 の (A)，(B) に示す．



(A) (B)

図 26: 各方式の比較

ランダム方式，max-remain 方式においては，全 WN 間での負荷の均衡が達成されていることが分かる．負荷の均等化に関する保証が全くないランダム方式においてさえ，負荷の非均衡が改善されていること，そして max-remain 方式

に大きく劣ることはないことから，動的負荷分散は送信元 WN 選択のオーバーヘッドが少ない方が有利であることが分かる．

そして，最も自然なアプローチである max-remain 方式において，最も遅く終了する WN における scan convert 時間が短い．本研究では，全 WN の scan convert が終了した後に最終合成を行う方法を採っているため，全処理時間が最も遅く scan convert を終了する WN によって決定されるため，max-remain 方式が最良であることが分かる．

次に，重なりの多い (B) において，bounding-box 方式では負荷の非均衡が改善されていない．これは，bounding-box 方式が重なりのある WN 間でのみ，セルの移送を行っており，重なりのある WN が終了した時点でセルの移送が行われない一方で，他の領域を担当する WN はまだ scan convert 処理を完了していないという状況が原因である．

以上の結果及び考察により，本評価においての最良である max-remain 方式と，bounding-box 方式を組み合わせることで，更なる性能向上が期待できる．すなわち，以下のようなアルゴリズムによって，WN_{idle} に対する送信元 WN WN_{from} を選択する．

- WN_{idle} と重なる領域を持つ WN の集合 dup_{idle} が存在し，なおかつ dup_{idle} の中にまだ scan convert を行っている WN の集合 $busy_{idle}$ が存在するならば，$busy_{idle}$ に対して max-remain によって WN_{from} を選択する
- そうでなければ，通常の max-remain 方式により，WN_{from} を選択する

この方式を max-bounding 方式と呼ぶことにし，max-remain 方式，bunding-box 方式との比較を行う．先ほどと同様の条件，対象で行った結果，(A) においてはほぼ変わらず，(B) において約 1% の性能向上が得られた．

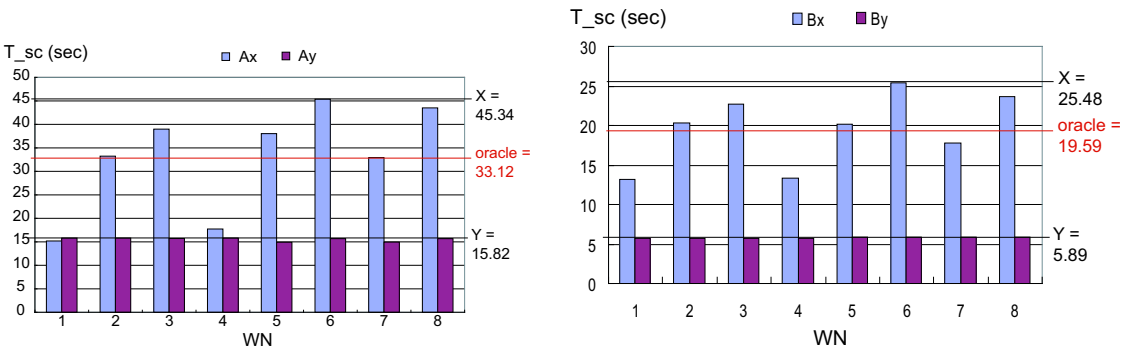
5.6 総合評価

本研究で述べた方法を全く適用しない，普通の並列 Cell Projection 方法 X と，これまでに述べてきた全ての知見を活用する方法 Y，すなわち

- 動的負荷分散は，max-bounding 方式で
- ERT，ERT 情報の共有を行う
- 最終合成には BSC を用いる

との比較を行う．

評価は，各 WN において，自身の持っていたセルと，動的負荷分散によって受信したセルの総 scan convert 時間 T_{sc} と，最終合成に要する時間 T_{cp} を計測することで行う．視環境は，(A)，(B) の両方を用い，それぞれのスクリーン解像度は，(A) が 900×300，(B) が 300×300 とする．各視環境での結果を，それぞれ図 27(A)，(B) に示す．



(A) (B)

図 27: 総合評価

X においては，各 WN での T_{sc} がほぼ T_{sc}の平均値 となっており，負荷の非均衡が改善されていることが分かる．また，5.5.2 で述べたように，全処理時間は最も遅く scan convert を完了した WN に決定される．したがって，本研究による成果である X は，Y と比べ，(B) において 4.32 倍の高速化を達成した．

また，Y において負荷の均等が理想的にとれている状態とは，全ての WN において $T_{sc} = T_{sc}$の平均値 となることである．X の平均値は Y の理想値の，1/2.65 となっており，これは ERT 情報の共有によってもらたされた，並列化ゆえの性能向上である．

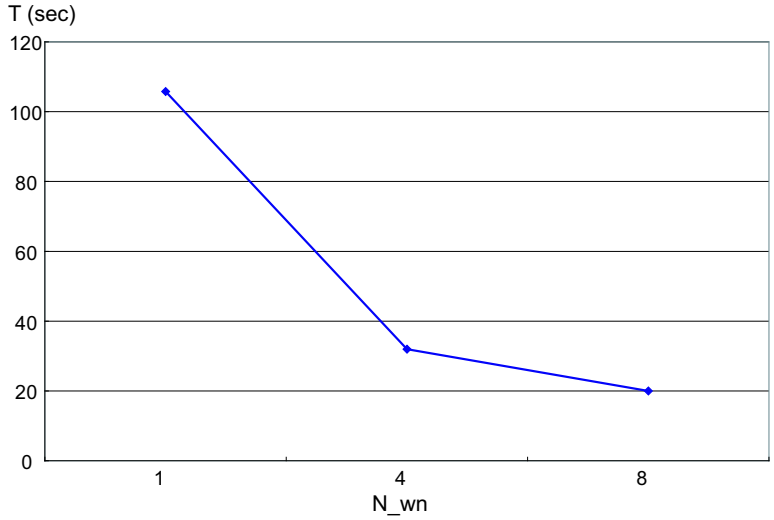
次に，WN 数 N_{WN} を変化させた場合の，全処理時間 T について評価を行う．評価条件は，次の通りである．

- 部分合成，ERT，ERT 情報の共有を行う
- 視環境は (B) で行い，スクリーン解像度は 900×300 とする

結果を，図 5.6 に示す．

$N_{WN} = 4$ において 3.31 倍，$N_{WN} = 8$ において 5.30 倍の性能向上を達成している．ここで，$N_{WN} = 1$ の場合が特に低速なのは，ERT 情報の共有が 1 台で

図 28: 台数効果



は行えないからである．

5.7 まとめと考察

本章では，負荷の非均衡が存在することの実証，提案する ERT-table 法及び ERT 情報の共有手法の評価，動的負荷分散の評価を行った．次に，性能面での考察を行う．

前章で述べた通り，本研究で行った実装では，30 万個のセルを 300×300 のスクリーン解像度で 8 台による並列 Volume Rendering を行うのに要する時間が 5.89sec である．同様のシステムである [8] では，50 万個のセルを 400×400 のスクリーン解像度で 8 台による並列 Volume Rendering を行うのに要する時間が 6.16sec あることを考えると，本研究で行った実装には改善の余地が残されていることが分かる．

本研究で行った実装でのボトルネックは，セルの scan convert である．この原因は，scan convert して生成された ray-segment を ray-segment リストに挿入する際，リストをたどる必要があることにある．したがって，ray-segment リストの管理を効率化することで高速化が期待できる．例えば，ray-segment リストを二分木やバランス木で管理するならば，ray-segment の挿入が通常のリストと比べ，$1/\log N$ で行える．このような改善と，プロセッサ台数を増加させることにより，第 1 章で述べたような実時間可視化環境に近づくことができると考えられる．

第6章　おわりに

本研究における成果は，次の通りである．

まず，非構造格子データを出力する数値シミュレーションとその結果の可視化を行う分散メモリ型並列計算機における，負荷の非均衡を指摘した．そして実際に評価を行った結果，確かに負荷の非均衡が存在することを示した．

次に，ソフトウェアで Cell Projection を行うライブラリを設計，実装し，並列化を施した．なお，このライブラリは，ソースコードを公開している．また，Cell Projection における ERT 手法を提案し，評価の結果，提案手法の有効性を示した．加えて，並列 Cell Projection での ERT 情報の共有手法を提案し，並列 Cell Projection の大幅な高速化を達成した．

そして，実装した並列 Cell Projection を用いて負荷の非均衡の存在を示し，動的負荷分散による解決策を提案した．動的負荷分散においては，いくつかの方針を提案し，評価を行った．最後に，提案手法を全て組み合わせた場合と，提案手法を全く使わない場合とを比較した結果，負荷の均等化を達成し，なおかつ ERT 情報の共有により，並列化特有の高速化を実現した．

今後の課題としては，以下が考えられる．

フレーム間での連続性

数値シミュレーション側は一定のタイムステップで計算結果を出力する．このとき，連続したタイムステップにおける負荷分散状況に大きな差はないことを，動的負荷分散に利用する．また，視環境の変化は通常連続的であり，次の視環境における可視化を行う際には，微小な差しかないことを，動的負荷分散に利用する．

OcTree 方式の改善

5.5.1 で述べたように，OcTree の scan convert 方法のオーバーヘッドを改善し，動的負荷分散方式として利用できるようにする．

台数を増やしての検証

本研究で利用した PC クラスタは 8 台のみである．より多いプロセッサ台数においても，本研究で述べた動的負荷分散方式，ERT 情報の共有手法が有効であるかを検証する．

ERT-table 法の改善

本研究で提案した ERT-table 法は，ERT-table の大きさを全て同じにして

処理を行っている．然るに，ボリュームデータには一般的に空間的局所性があり，ERT-table の大きさが一定では，可視化すべきデータが存在しない領域，データが密集している領域を等しく扱うため，効率が悪い．したがって，ERT-table の大きさを，QuadTree(四分木) などを用い，適応的に変化させることによって性能向上が期待できる．

scan convert の改善　ray-segment リストを二分木やバランス木を用いる実装を行うことにより，scan convert の大幅な高速化が期待できる．

謝辞

本研究の機会を与えてくださった富田眞治教授に深く感謝の意を表します．また，日頃から有用な議論をして頂いた，富田研究室各位に感謝いたします．

本研究で用いた非構造格子データをご提供いただいた北陸先端科学技術大学院大学 松澤 教授，渡邊氏に慎んで感謝いたします．

参考文献

[1] 山本 恭弘他: 有限要素法を用いた心臓大動脈の触診シュミレーション, 日本バーチャルリアリティ学会第 6 回大会論文集 (2001).

[2] Levoy, M.: Display of Surfaces from Volume Data, *IEEE Computer Graphics and Applications*, Vol. 8–3, pp. 29–37 (1988).

[3] Max, N., Hanrahan, P. and Crawfis, R.: Area and Volume Coherence for Efficient Visualization of 3D Scalar Functions, *Computer Graphics (San Diego Workshop on Volume Visualization)*, 5, Vol. 24, pp. 27–33 (1990).

[4] Lacroute, P. and Levoy, M.: Fast volume rendering using a shear-warp factorization of the viewing transformation, *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM Press, pp. 451–458 (1994).

[5] Westover, L.: Interactive volume rendering, *Proceedings of the 1989 Chapel Hill workshop on Volume visualization*, ACM Press, pp. 9–16 (1989).

[6] Chen, L., Fujishiro, I. and Nakajima, K.: Parallel performance optimization of large-scale unstructured data visualization for the earth simulator, *Proceedings of the Fourth Eurographics Workshop on Parallel Graphics and Visualization*, Eurographics Association, pp. 133–140 (2002).

[7] Muraki, S., Lum, E. B., Ma, K.-L., Ogata, M. and Liu, X.: A PC Cluster System for Simultaneous Interactive Volumetric Modeling and Visualization, *IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, pp. 95–102 (2003).

[8] Ma, K.-L. and Crockett, T. W.: Parallel visualization of large-scale aerodynamics calculations: A case study on the Cray T3E, *IEEE Parallel Rendering Symposium*, pp. 95–104 (1997).

[9] Porter, T. and Duff, T.: Compositing digital images, *Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, ACM Press, pp. 253–259 (1984).

[10] 松井学, 竹内彰, 伊野文彦, 荻原兼一: 累積不透明度の伝播による並列ボリュームレンダリングの計算量削減, 信学技報, 電子情報通信学会, pp. 13–18 (2003).

[11] Fuchs, H., Abram, G. D. and Grant, E. D.: Near real-time shaded display of rigid objects, *Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, ACM Press, pp. 65–72 (1983).

[12] Ma, K.-L., Painter, J. S., Hansen, C. D. and Krogh, M. F.: Parallel volume rendering using binary-swap compositing, *IEEE Computer Graphics and Applications*, Vol. 14, No. 4, pp. 59–68 (1994).

[13] 對馬 雄次他: ボリュームレンダリング専用並列計算機 ReVolver のアーキテクチャ, 情報処理学会論文誌, Vol. 36–7, pp. 1709–1718 (1995).

[14] Levoy, M.: Efficient Ray tracing of volume data, *ACM Transactions on Graphics*, Vol. 9–3, pp. 245–261 (1990).