セル投影型並列ボリュームレンダリングの Early Ray Termination による高速化

高山征大[†,☆] 森眞一郎[†]
中島康彦[††] 富田眞治[†]

本稿では，セル投影法を用いた並列ボリュームレンダリング処理に，従来効率的な適用が困難とされてきた Early Ray Termination（**ERT**）による最適化を適用する手法，ならびに，並列処理環境において各計算ノードでの ERT の進捗状況を共有することで複数ノード間にまたがる ERT 処理を実現する手法を提案する．動的負荷分散機能を実装したセル投影型並列ボリュームレンダリングシステムに提案手法を実装し，307,565 個の要素からなる非構造格子データを，8 台の PC クラスタを用いて可視化を行った結果，提案手法を用いない場合に比べて数倍の性能向上を達成できた．

Cell-projection Parallel Volume Rendering with Early Ray Termination

Motohiro Takayama,[†,☆] Shinichirou Mori,[†]
Yasuhiko Nakashima[††] and Shinji Tomita[†]

A cell-projection parallel volume rendering system for unstructured grid volume data is proposed in this paper. For this system, the authors have modified early ray termination (ERT) scheme so that it could efficiently prune the cell-projection of invisible cell. The authors have also proposed a scheme to improve the efficiency of ERT among processing nodes by weak sharing of ERT status of each node. In order to alleviate load imbalance due to view-dependency in scan-conversion and the dynamic behavior of early ray termination, the authors have also implemented dynamic load balancing mechanism into their system. Preliminary evaluation of this system shows up to 4.32-times performance improvement compared to the system without early ray termination and dynamic load balancing.

1. はじめに

我々は，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向けて，個人あるいは小規模な組織単位で占有利用可能な PC クラスタ☆☆ を用いて，インタラクティブな数値シミュレーションおよびその可視化を実時間処理する環境について研究を行っている[1)~3)]．

本稿では，現在開発を行っている実時間可視化環境のうち，セル投影法を用いた非構造格子データの並列ボリュームレンダリング処理（以下，**PVR** と呼ぶ）に対して，効率的な実装が従来困難とされてきた Early Ray Termination（以下，**ERT** と呼ぶ）[4)] による最適化を効率良く適用する手法を提案する．

なお，以下では議論を簡単化するため対象とする非構造格子データ（セル）の形状は四面体を仮定するが，提案手法は階層構造を持つものを含め任意の非構造格子に適用可能である．また，セルの光学モデルとしては，各頂点に与えられた属性値と視点位置に応じて変化するセルの視線方向の厚みの両方を考慮して透明度を補正（補間）する光学モデルを採用する[5)]．

大規模データの実時間可視化において，不可視部分の処理を省略する Visibility Culling は必要不可欠な技術である[6)]．Visibility Culling のうち，特に視点から見て前方にある物体によって遮蔽される後方物体の描画を省略する技術は Occlusion Culling と呼ばれる．不透明ポリゴンデータのサーフェイスレンダリング（等値面表示を目的としたボリュームレンダリングも含む）においてはポリゴンデータ間の幾何学的な位置関係のみからスキャン変換前に不可視判定が可能であり容易に Occlusion Culling を実装することが可能で

† 京都大学大学院情報学研究科
Graduate School of Informatics, Kyoto University
†† 京都大学大学院経済学研究科/JST
Graduate School of Economics, Kyoto University/JST
☆ 現在，株式会社東芝
Presently with TOSHIBA Corporation

☆☆ ノード数 100 程度までを想定

ある．ところが，半透明データの可視化を行うボリュームレンダリングにおいては，幾何学的な位置関係のみでは物体間の遮蔽関係が一意に定まらず，ピクセル単位の透明度に基づいた不可視判定が必要となる[5),7)]．

規則構造格子データのボリュームレンダリングでは，視点位置が与えられると全ピクセルに対して共通なデータ間の前後関係（全順序関係）を定義することが可能である．この性質を利用することで，視線に沿って計算した透明度に基づいた Early Ray Termination（ERT）によるピクセル単位の不可視判定処理が可能である．非構造格子データに対するレイキャスティング型のボリュームレンダリングアルゴリズムでも，ピクセルごとに隣接関係にあるセルを追跡するため ERT による不可視判定が可能である．しかしながらデータサイズが大きくなるとレイキャスティング型は隣接関係の追跡処理自体のコストが非常に高くなるという問題点がある．並列化という観点ではピクセル単位の並列性が抽出可能であるが，大規模データが分散配置された状態での並列化を考えるとプロセッサ間にまたがる視線追跡のオーバヘッドは無視できない．そのため，大規模非構造格子データの並列可視化においては，セルの隣接関係を追跡する必要がなく，セル単位の並列性が抽出可能なセル投影型のボリュームレンダリングアルゴリズムが多く用いられている．ところが，セル投影型ではセルの処理順序が必ずしも視線に沿った前後関係と一致するとは限らない[7),8)]．また，1 回の投影処理で複数のピクセルに影響を与えるため，ピクセル単位の透明度に基づいた不可視判定を行う ERT と必ずしも親和性が高いとはいいがたい．

そこで本稿では，1) 部分累積透明度を用いたピクセル単位の不可視判定と，2) スクリーンをいくつかの領域に分割したサブスクリーン単位の不可視判定，を併用することで ERT の効率的な実装を行うとともに，ERT 情報の緩い共有により並列化にともなう ERT 効率の低下を軽減する手法を提案する．

2. セル投影型並列ボリュームレンダリング

2.1 セル投影法

セル投影法[9)] でのボリュームレンダリング処理は以下の 3 つのフェーズで構成される（**図 1** 参照）．

(1) 投影（projection）：まず，与えられた三次元データをスクリーンに投影する．

(2) スキャン変換（scan conversion）：次に，投影された個々のセルに対して，スキャン変換処理を行う．具体的には，セルが投影されたスクリーン上のピクセル $P(x, y)$ を通る視線がセルに入射後，通過するまでに，$P(x, y)$ に与える寄与値（色値（RGB），および不透明度（α））と，入射点と通過点の奥行き座標値（それぞれ z_{front}，z_{back}）を求める．以下では，これら 4 つのパラメータで構成されるデータを **ray-segment** と呼ぶ．その結果，個々のセルが投影されるスクリーン上の各ピクセルに対して 1 つの ray-segment が計算される．

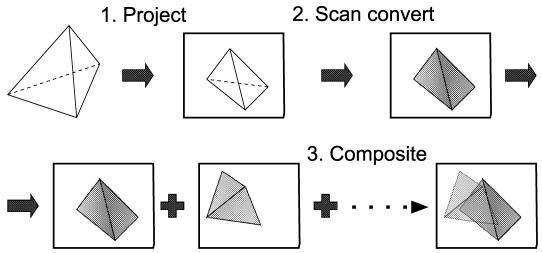


図 1 セル投影法を用いたボリュームレンダリング処理
Fig. 1 Cell-projection scheme.

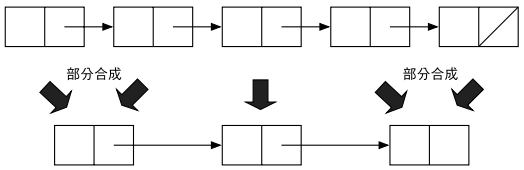


図 2 部分合成による ray-segment-list の縮小
Fig. 2 Partial composition.

このスキャン変換処理を全セルに対して行った結果得られた ray-segment を，対応するピクセルごとに分類し，かつ，視点に近いものから遠いものへの順番（以後，「奥行き順」と呼ぶ）に並べたリストを作成する．

(3) 合成（composition）：全セルのスキャン変換処理が完了した時点で，各リストで奥行き順に並べられた ray-segment を手前から順次合成することで，スクリーン上のピクセル値を得る．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．

なお，スキャン変換処理によって作成されたリストにおいて，隣接する ray-segment のうち，一方の通過点と他方の入射点が一致するものは奥行き順で畳み込み合成する（これを**部分合成**と呼ぶ）ことが可能である（**図 2**）．部分合成した結果もまた ray-segment となり，結果として ray-segment-list を縮小することができる．したがって，スキャン変換時に新しく ray-segment をリストに挿入する際，可能であれば部分合成を行ったうえでリストに挿入することで，リストの長さを短くするという最適化が可能である．また，部分合成を行った後の ray-segment の不透明度は，合成

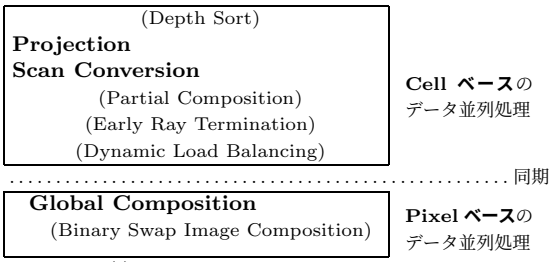


図 3 並列プログラム実装の概要
Fig. 3 Parallel implementation.

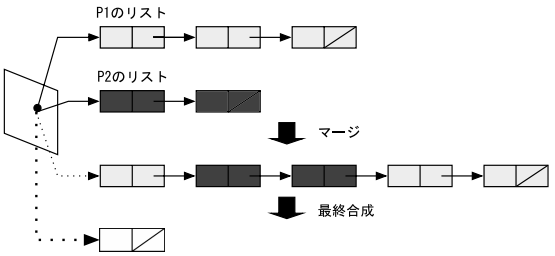


図 4 ray-segment-list のマージ，最終合成
Fig. 4 Composition of ray-segment-lists.

前の 2 つの ray-segment の不透明度より小さくなることはない．この性質を利用することで後述の ERT の効率を上げることが可能である．

2.2 セル投影法の並列化

我々の並列プログラム実装では，シミュレーション結果（中間結果も含む）が得られた時点でただちに可視化を行うインタラクティブなシミュレーション環境の構築を目標としているので，可視化対象となるデータ（セル）の初期配置は当該シミュレーション結果を生成したプロセッサに分散して配置されている状態そのものを初期配置とする．また，システムの規模としては，100 台規模の PC クラスタ環境までのスケーラビリティを前提としている．

セル投影法では，投影処理とスキャン変換処理ではセル単位の並列性が利用可能であり，一方合成処理ではピクセル単位の並列性が利用可能である．現在の我々の並列実装では，セル単位の並列処理を行う第 1 フェーズとピクセル単位の並列処理を行う第 2 フェーズに分けて，異なる並列処理形態でそれらを順次実行する．**図 3** に並列プログラム実装の概要を示す．

第 1 フェーズでは，各プロセッサは割り当てられたセルを投影ならびにスキャン変換し，自身の ray-segment-list を構築する．全プロセッサが各々に割り当てられたセルをすべて scan convert し終わると第 2 フェーズの実行を開始する．ここでは，全プロセッサの ray-segment-list を，スクリーン上のピクセルごとに 1 つの ray-segment-list 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 1 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 4**）．ray-segment-list のマージならびに最終合成では，スクリーンを複数の領域に分割し，各プロセッサに担当領域を割り当てて並列処理を行う．

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-list 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が図れる．

本研究で採用する並列化アルゴリズムは基本的には文献 10) で提案された方式に準じたものであるが，文献 10) とは異なり，スキャン変換処理と最終合成処理の同時実行は行っていない．

3. セル投影法への ERT の応用

Early Ray Termination（ERT）は，ボリュームレンダリング処理の高速化手法の 1 つで，レンダリングの対象の性質やレンダリング時に使用する伝達関数に依存するが，高速化にきわめて有効であることが知られている[4]．ERT はレイキャスティング型ボリュームレンダリングにおいて多く用いられてきた手法であり，ピクセルごとに計算されるボクセルの寄与値累積計算過程において，ある点までの累積計算結果として得られる不透明度がある閾値を超えた場合（以下では，この状態を **terminated** と呼ぶ）に，その点より奥にあるボクセルからの寄与は十分小さいと考え，それ以降の累積計算を打ち切ることで高速化を図る手法である．

今，セル投影型 PVR への ERT の応用について考えてみる．本来 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ため，セル投影型 PVR において ERT を活用できるのは単純に考えると ray-segment の合成処理だけとなる．しかしながら，セル投影型 PVR に要する時間のうち，大部分を占めるのはスキャン変換処理である．したがって，スキャン変換処理を省くことができなければセル投影型 PVR において本質的な高速化につながらない．

そこで我々は，各ピクセルの ERT に関する履歴に基づき，サブスクリーン単位でスキャン変換処理自体の省略可能性判定を行う Conservative Subscreen Termination（CST，保守的 ST）法を提案し，さら

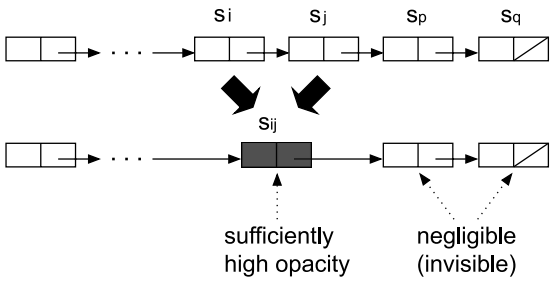


図 5 部分累積不透明度による ERT
Fig. 5 ERT using partially accumulated opacity.

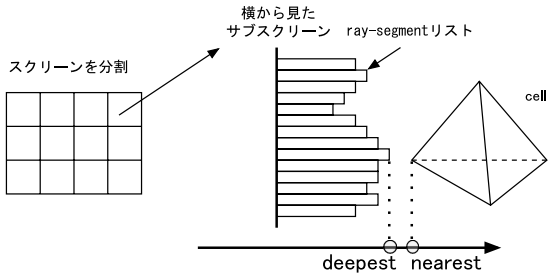


図 6 Conservative Subscreen Termination 法
Fig. 6 Conservative subscreen termination scheme.

に，並列処理への応用について検討を行った．

3.1 Conservative Subscreen Termination

3.1.1 提案手法の概要

いま，**図 5** のようなあるピクセルにおける ray-segment-list 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 は処理を省略することが可能である．この省略可能性判定（以下 ERT 判定と呼ぶ）は s_i よりも前にある ray-segment の状態が未知であっても行うことが可能である．また，s_{ij} の不透明度が十分不透明でなかった場合でも，s_j より前にある既知の ray-segment 群がすべて部分合成可能であると仮定して仮想的に計算した部分累積不透明度が十分不透明な場合には，s_j よりも後ろに生成される ray-segment は処理を省略することが可能である．このように，あるピクセルでの ERT 判定は，厳密な累積不透明度が未知であっても，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 のスキャン変換処理を省略できる場合がどのような場合か考える．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，それまでにスキャン変換された，c よりも前方に存在するセルを変換した ray-segment-list が存在し，なおかつそれらのリストすべてが terminated になっているならば，c の投影された領域はすでにすべて terminated な状態であることになる（**図 6**）．したがって，この場合に c のスキャン変換処理は省略可能である．

しかし，このようにピクセル単位で ERT 判定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 判定を行うには，スキャン変換を行うのと同様の反復を繰り返さなくてはならず，ERT 判定のためのオーバ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 判定を行うのではなく，セルを囲む程度の大きさのサブスクリーン単位で ERT 判定を行う方法を提案する．すなわち，各セルごとに，投影領域に対応する全ピクセルに対して ERT 判定を行う代わりに，1) セルを囲むサブスクリーン内の全ピクセルが terminated に達していること，ならびに，2) 当該サブスクリーン内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*）との関係が $deepest < nearest$ であること，の 2 つの条件を判定することにより ERT 判定を行う．

このように，ピクセル単位ではなく，各セルの投影面が属するサブスクリーンを単位として保守的に ERT 判定を行うことから，本手法を Conservative Subscreen Termination（CST，保守的 ST）法と呼ぶ．保守的 ST 法は，その実装において ERT の情報を表で管理する点，部分累積不透明度を活用する点では松井らの論文[11),12)] と同様であるが，適用対象とするボリュームデータの形状（構造），レンダリングのアルゴリズム，ERT 適用の方法が異なっている．

3.1.2 利点・欠点

この方法では，あるセルの ERT 判定に要する処理は，1) セルを囲むサブスクリーンを見つけること，2) 当該サブスクリーン内の全ピクセルが terminated に達しているかの 1 bit の真理値を調べること，3) サブスクリーンの Z 値とセルの Z 値の比較，といった単純な処理だけで済み，ERT 判定に要するオーバヘッドがきわめて少ない．

一方で，サブスクリーン内の全ピクセルが terminated でないとセルのスキャン変換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 判定を行う場合と比較して減少する．また，Z 値が大きな ray-segment が先にスキャン変換されてしまうと，その ray-segment よりも手前にあるセルのスキャン変換が省略できない状況も発生する．ただし，後者に関してはセルをあらかじめ大雑把にソートしておくことで状況を改善することは可能である．

このように提案手法では，ERT 判定を緩めに（保守的に）行うことで ERT 判定のオーバヘッドを軽減し，これによりスキャン変換時における ERT 判定を可能とすることでセル投影型 PVR の高速化を図っている．

なお提案手法は，あるサブスクリーンに着目すると，Z 値が *deepest* に相当する位置にスクリーンへの投影面積が当該サブスクリーンと一致する不透明半平面（あるいは半曲面）を仮想的に設置して，その面によって遮蔽されるセルに対して Occlusion Culling を行っていると見なすことも可能である．

3.2 ERT 情報の共有

セル投影型並列 Volume Rendering に単純に保守的 ST 法を組み合わせただけでは，ERT による効果が十分には発揮できない．これは，スキャン変換が各プロセッサで独立かつ並行して行われており，あるピクセルが terminate 状態になったことが他のプロセッサに伝わらず，本来省略できる領域について処理を継続して行うことによる無駄が生じるためである．

そこで本稿では，各プロセッサが ERT 情報の更新を非同期かつ独立して行うことを基本としつつ，定期的に各プロセッサの ERT 情報を交換することで ERT 情報を共有する手法を提案する．以下，提案手法の一実装例について説明する．この実装では簡単化のため，ERT 情報を管理する管理プロセッサを導入する．

まず，各プロセッサが保守的 ST 法を用いて ERT 処理を非同期かつ独立して行うために，それぞれのプロセッサにローカルな ERT-table を持たせる．適当な周期で管理ノードの ERT-table の情報と各プロセッサにローカルな ERT-table の情報を比較し ERT-table の情報を最適な状態に更新する☆．

具体的には，ERT-table の各々にエントリに対して，各プロセッサから集めた当該エントリのデータのうち *deepest* が最小であり，かつ *terminate* しているデータを見つけ，その値を用いて当該エントリを更新する．これにより，最も手前で terminate している ERT-table エントリの情報を全プロセッサで共有することができる．この際，実際に通信する ERT 情報は，ERT-table の 1 エントリあたり terminate しているか否かを表す bool 値（1 bit），と *deepest* 値のみであり，少ない通信量で ERT 情報の共有を実現することができる．

1 セルのスキャン変換が終了するたびにプロセッサ間で ERT 情報を交換すると ERT-table の情報は真に最新のものとなるが，そのために必要なコストはキャッシュの一貫性制御をハードウェアで行う共有メモリシステムにおいてもきわめて大きく，ERT-table を最新に保つことで得られる ERT の効果より大きい．本実装において，ERT 情報の交換を「適当な周期」で行うのはこのためである☆☆．我々は，この適当な周期での ERT 情報の共有を，ERT 情報の弱共有（Weak Sharing of ERT Information）と呼ぶ．

4. 評　　価

本章では，動的負荷分散機能を実装したセル投影型並列ボリュームレンダリングシステムに提案手法を実装し，その評価を行った結果を示す．

4.1 評 価 環 境

評価に用いた三次元データは，正常者の動脈弓における血流シミュレーション結果のデータであり，各セルの形状は四面体の非構造格子である．セル数は 307,565，頂点数が 62,475 であり，各頂点はそれぞれ圧力値 $P\,(\mathrm{N/m^2})$，速度ベクトル $v\,(\mathrm{m/s})$ を浮動小数点型の数値データとして持つ．この速度データを可視化した例を，**図 7** に示す．伝達関数の不透明度は，全ノードの不透明度の平均値が 0.3 となるように設定した．また，各プロセッサへのデータの初期配置は，セル数がほぼ均等であり，かつ，できる限り近接するセルが同一プロセッサに配置されるように設定した．**図 8** は各プロセッサに割り当てられているセルを，Cell Projection 法によって各々のプロセッサで可視化した結果である．

図 9 は，2 つの直交する視点 (A)，(B) から見た，初期配置されたセルのバウンディングボックス表現である．個々の長方形が各プロセッサ内のセルのバウンディングボックスに対応する．なお，以下では特に明記しない限り，視点 (A) および (B) それぞれに対してスクリーンサイズは 900×300 および 300×300 で

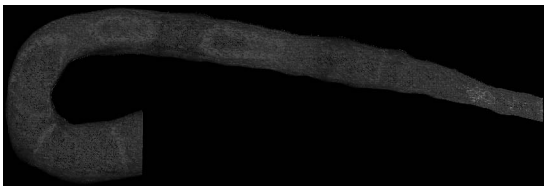


図 7 動脈弓データ
Fig. 7 Sample data (Human Aorta).

☆ 本来は重なる領域を持つプロセッサ間のみで ERT 情報を共有すればよく管理ノードは不要である，現在は実装の簡単化のため全プロセッサで ERT 情報を共有している．

☆☆ 「適当な周期」とは，情報共有のコストがそれによって得られる ERT の効果より十分小さくなるような周期であり，PVR を実装するハードウェア環境で定まる周期である．

図 8 各プロセッサが持つセル
Fig. 8 Initial cell assignment.

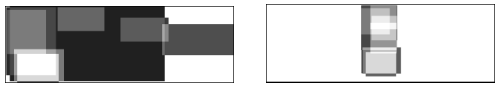
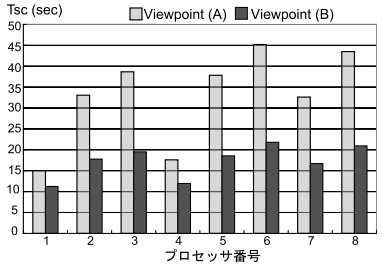
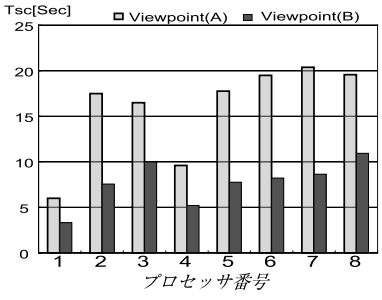


図 9 視環境
Fig. 9 View directions.



(a) ERT なし



(b) ERT あり/共有なし

図 10 各プロセッサでのスキャン変換時間（動的負荷分散なし）
Fig. 10 Scan conversion time on each WN (w/o DLB).

ある．

使用した計算機環境は，8 ノードの PC クラスタ（Pentium4 3 GHz，主記憶 1 GB，1 GbE 接続，Linux）と 1 台の管理ノード（Pentium4 2 GHz，1 GbE 接続）である．通信ライブラリには LAM/MPI を使用した．予備評価として，ERT ならびに動的負荷分散をともに行わない状態での各プロセッサのスキャン変換時間 T_{sc}（図 3 の第 1 フェーズに相当．以下同じ）を**図 10** (a) に示す．セル数がほぼ同一であるにもかかわらず，プロセッサ間で負荷の不均衡が起こっていることが分かる．また，この不均衡の度合いは視点に依存しており，視点 A の場合が不均衡の度合いが高いことが分かる．図 10 (b) は，各プロセッサで ERT を適用するが ERT 情報の共有は行わない場合である．ERT を適用する場合の閾値はすべての実験で 0.9 とした．ERT を適用することで，各プロセッサのスキャン変換時間が短縮でき，ほぼ 2 倍の性能が得られていることが分かる．さらに，プロセッサ間の負荷の不均衡のパターンが，ERT の適用の有無で変わることも確認できる．これはすなわち，ERT を適用した場合には視点情報のみに基づく動的負荷分散のみでは，十分な負荷の均衡が得られないことを示している．

4.2 逐次処理環境における ERT の効果

本節では，ERT によるスキャン変換処理の省略の効果について検証するため，1 台のプロセッサですべてのセルを処理する場合について実験を行う．提案手法の実装においては，ERT-table サイズ，ERT 更新頻度等の実装パラメータが存在する．そこでこの 2 つのパラメータが ERT 効率にどのような影響を与えるかを以下で調査する．

4.2.1 ERT-table サイズと ERT 効果の関係

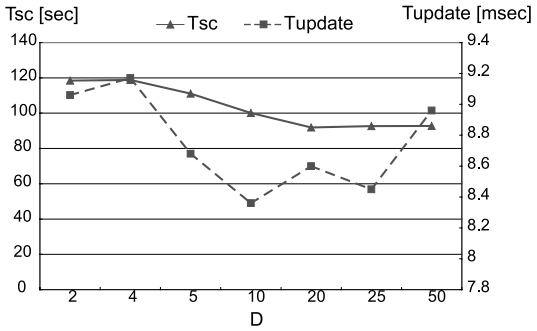
ERT-table のサイズはスクリーンの分割数によって決定される．ここでは，スクリーンの一辺の分割数 D を変化させたときの，1) 全セルをスキャン変換し終えるまでの時間 T_{sc}，2) 全 ERT-table を一度更新するのに要する時間 T_{update}，ならびに，3) ERT によってスキャン変換を省略できたセル数 N，について評価を行う．測定条件としては，ERT-table の更新間隔は残りセル数の 1/10 間隔とし（次節参照），部分合成は行っていない．

図 11 (1)，(2) は ERT の効果が顕著である視環境 (B) に対する実験結果である．ともに横軸は分割数 D であり，(1) の左縦軸は T_{sc} (sec) を，右縦軸は T_{update} (msec) を，(2) の縦軸は N（個）である．

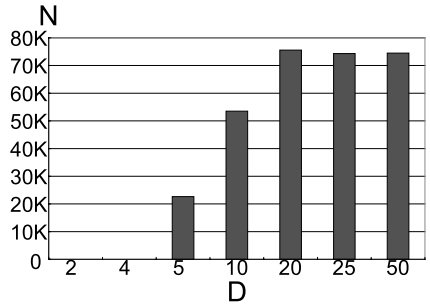
このデータおよび視環境では，$D = 20$ のとき，すなわち ERT-table の 1 エントリに対応する領域の幅と高さが 15 のときに，T_{sc} が最も小さく，かつ N が最も大きくなり最良である．このとき，全セルの約 1/4 のセルのスキャン変換が省略可能である．また，ERT を適用しない場合のスキャン変換時間は約 170 秒であったので，約 1.9 倍の速度向上を達成していることが分かる．一方で，T_{update} は 3 番目に小さな値となっている．

なお現在の実装では簡単化のために，セルの投影領域が 1 つの ERT-table エントリに対応する領域内に収まらない場合は ERT の適用を行っていない．そのため，分割数をある程度以上大きくすると ERT により省略できるセルの数が減少していることが分かる．

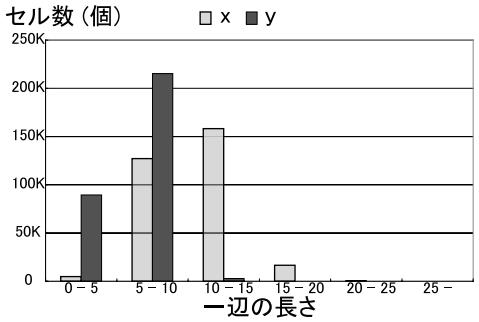
図 11 (3) は，投影されたセルの x，y 方向の最大長



(1) スキャン変換時間 T_{sc} と ERT-table 更新時間 T_{update}



(2) 省略できたセル数 N



(3) セルの大きさの分布

図 11 ERT-table サイズと ERT 効果
Fig. 11 The influence of ERT-table size on ERT effect.

の分布図である．この図からセルの投影領域サイズが 15×15 の範囲にほとんど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．

これらの結果から，D の設定に際しては，前処理の一環としてセルの大きさに関する平均と分散を求め，その情報をもとに D の値を設定すればよいと考えらる．

4.2.2 ERT-table の更新頻度と ERT 効果の関係

提案手法の実装においては，1 セルのスキャン変換を行うたびに ERT-table の更新を行うのではなく，複数セルのスキャン変換後にまとめて ERT-table の更新を行う方式をとっている☆．また，その更新間隔自体も残りセルの数に応じて動的に変化させる方式をとっている．具体的には，次の ERT-table 更新までの間隔を (その時点における全セル数)/更新頻度定数 C としている☆☆．

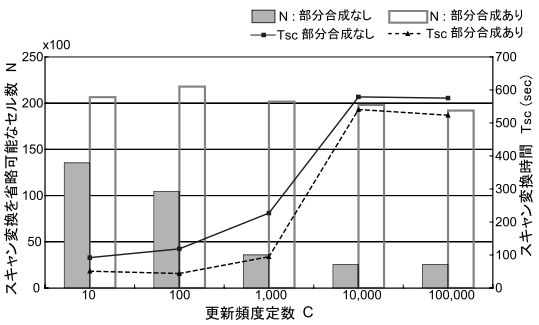


図 12 ERT-table の更新頻度と ERT 効果
Fig. 12 The influence of ERT-table update frequency of ERT effect.

本節では，更新頻度定数 C を変化させた場合における，1) 全セルをスキャン変換し終えるまでの時間 T_{sc} ならびに，2) ERT によってスキャン変換を省略できたセルの換数 N，について，部分合成有無の両方の場合について評価を行う．ERT-table の大きさは前節の実験結果より $D = 20$ としている．**図 12** に実験結果を示す．横軸は更新頻度定数 C であり，縦軸は N および T_{sc} である．

予想に反して更新頻度定数 C が大きい場合，すなわち短い間隔で更新する方がスキャン変換を省略できたセル数が少なくなっている．また，スキャン変換時間も増加している．スキャン変換時間増加の第 1 要因としては，頻繁に ERT-table を更新することによる ray-segment-list 検索回数の増加があげられる．第 2 要因としては，頻繁に ERT-table を更新することによる *deepest* 値の増加が考えられる．本来ならば ray-segment-list の前の方で terminated に達するにもかかわらず一時的な *deepest* 値の増加により省略

☆ これは，ERT-table の更新処理時には，各 ray-segment-list をたどって部分累積透明度を計算する必要があり，1 セルのスキャン変更ごとに行うとオーバヘッドが高いためである．

☆☆ 処理の進行とともに ray-segement-list が terminated になる確率が次第に高くなるため．

不可能なセルが発生する可能性があるためである☆. $C > 1{,}000$ ではこのオーバヘッドのため，ERT を適用しない場合（$T_{sc} = 170$ [sec]）よりも速度が低下している．

しかしながら，部分合成を行うことでこれらの問題が緩和されていることも確認できる．特に，スキャン変換を省略可能なセル数は部分合成を行うことで大幅に増加することが分かる．また，T_{sc} に関しては C が 1,000 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．今回の実験では，$C = 100$ が最適であることが分かる．

今回得られたデータのみから，C の値に関する普遍的な決定法について結論づけるのは困難であるが，今回の傾向を見る限り仕事量（残りセル数）の 1%から 10%を処理するごとに ERT-table を更新すればよいと考えられる．ただし，セルの不透明度が高いことがあらかじめ分かっている場合には，更新頻度をやや高めに設定すると ERT の効果が上がることが定性的に期待できる．

4.2.3 総合評価

以上の結果をもとに，視環境 (A)，(B) に対して ERT の適用の有無によるスキャン変換時間削減の効果をまとめると**表 1** のとおりとなる．ERT を行う場合は，前項までで得られた知見からスクリーン分割数 $D = 20$，更新頻度定数 $C = 100$ とし部分合成を適用した結果である．なお，視環境 (A) に関してはスクリーンサイズを 300×100 として実験を行った☆☆.

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，図 12 から分かるように約 2/3 のセルを不可視と判定でき，その結果 3.86 倍の高速化を達成できている．

表 1 ERT によるスキャン変換処理時間の削減
Table 1 The effect of the CST in sequential execution.

ERT の有無	なし	あり
視環境 (A)	49.34	39.06
視環境 (B)	169.87	43.99

（単位：sec）

☆ Subscreen の *deepest* 値に影響を与えるある視線に関して，視点に近い順に A，B，C，D の 4 つのセルが寄与する場合を考える．このとき，B と D のセルの不透明度は閾値よりも十分に高いと想定する．今，処理の順番が A，D，B，C で，ERT-table の更新頻度が 2 の場合，D の不透明セルの Z 値が *deepest* として登録される．後続の B，C は D よりも前方にあるためスキャン変換が必要となる．もし更新頻度が 3 であった場合，B の不透明セルの Z 値が *deepest* として登録されるため，B よりも後方にある C のスキャン変換は省略が可能となる．このような状況が発生すると省略不可能なセル数の増加が観測される．なお，今回の実験において $C = 100{,}000$ の場合，更新頻度は 3 から 1 の間で変化している．

☆☆ メモリ不足のため 900×300 ではプロセッサ 1 台で実行できなかったためである．

4.3 並列処理環境における ERT の効果

本節では，ERT を並列処理環境で適用した場合における，ERT 情報共有の効果，ならびに，動的負荷分散の効果について検証を行う．ERT-table 構築のためのスクリーン分割数 D は 20，各プロセッサでの ERT-table 更新頻度定数 C は 100 とし，特に明記しない限り部分合成を行う．並列 Cell Projection に使うプロセッサ台数は 8 台である．なお，最終合成には文献 13) で提案されている Binary Swap Image Compositon を適用しており，いずれの視環境においても 1 秒以下で合成を完了している☆☆☆.

4.3.1 ERT 情報の共有によるスキャン変換処理の削減

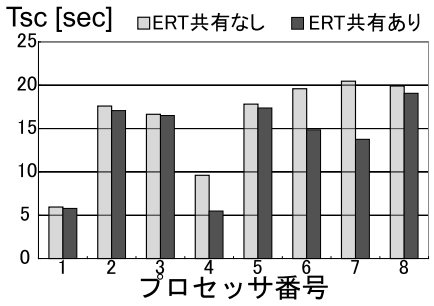
まず，ERT 情報の共有によるスキャン変換処理の省略の効果について検証する．各プロセッサでの ERT の効果を調べるため動的負荷分散は行わない．

基本データとして，ERT 情報の共有を一度行うのに必要な時間 T_{share} を計測したところ，視環境 (A) では $T_{share} = 2.5$ msec，視環境 (B) では $T_{share} = 0.6$ msec となった．現在の実装では，ERT 情報の共有は各プロセッサでの ERT-table 更新と同じタイミングで行っており，表 1 の ERT なしの場合のスキャン変換時間から 1 セルあたりの平均のスキャン変換時間を逆算して計算すると，ERT 情報共有を導入することによる情報共有自体のオーバヘッドは全スキャン変換時間の 10%未満であることが分かる．

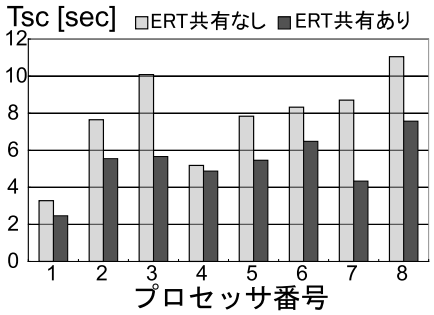
次に，ERT 情報の共有を行う場合と行わない場合の比較を行った結果を**図 13** に示す．横軸はプロセッサ番号，縦軸は各プロセッサでのスキャン変換時間 T_{sc} (sec) である．

プロセッサ間のバウンディングボックスの重なりの少ない視環境 (A) においては，わずかに領域が重なっているプロセッサ 1 と 2，ならびに，プロセッサ 6 と 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，バウンディングボックスの重なりの多い視環境 (B) においては，全プロセッサにおいて大幅な性能向上が見られる．なかでも，プロセッサ 7 に関しては約 2 倍の高速化を達成している．

☆☆☆ 現在の我々の実装においてはまだ最終合成はボトルネックにはなっていないが，プロセッサ数の増加等にともない最終合成処理がボトルネックとなる可能性は否定できない．このような状況下での，最終合成の高速化に関しては，我々も含め多くの研究が別途行われている[2),14)〜16)].



視環境 (A)



視環境 (B)

図 13 ERT 情報の共有によるスキャン変換処理の削減
Fig. 13 The effect of the ERT information sharing.

なお，今回の実験は管理ノードを含めて 9 台のプロセッサを用いての実験であるため 1 回の情報共有に必要な 2.5 msec や 0.6 msec という値は問題とならない．しかしながら，100 台規模のシステムを考えると管理ノードに情報を集中する現在の実装では管理ノードの処理がボトルネックとなる．そのため，必要な情報のみをノード間で局所的に共有するメカニズムの開発が必須である．

4.3.2 動的負荷分散の効果

ERT ならびに ERT 情報共有を適用することでスキャン変換時間を短縮できることは確認できたが，図 13 に示すように個々のプロセッサの処理時間には大きな不均衡が起こっている．そこで，動的負荷分散を行うことで負荷の均衡を試みた．動的負荷分散の機能としては，各プロセッサが自分に初期配置されたセルのスキャン変換を終了した時点で，未処理のセルが最も多いプロセッサから，残りセル数の 1/3 を貰って処理を行う方式を採用した[17),18)]．その結果，プロセッサ間の負荷の不均衡が著しく改善され，最も処理時間の遅いプロセッサでのスキャン変換時間は視環境 (A) および (B) で，それぞれ 15.82 sec ならびに 5.89 sec となり全体の処理時間がさらに改善された．このとき，各プロセッサ間での負荷の不均衡は最大でも 7%以下に抑えることができた．

表 2 並列処理環境における最適化効果
Table 2 Overall effects.

Scan Conversion Time T_{sc} [sec]		
Optimization Level	(A)	(B)
Base	45.37	25.48
DLB(*)	33.12	19.59
ERT	20.38	10.93
ERT + SHARE	19.06	7.57
ERT + DLB	18.40	8.87
ERT + DLB + SHARE	15.82	5.89

ERT: Early Ray Termination
SHARE: ERT Information Sharing
DLB: Dynamic Load Balancing
(*): Estimated based on “Base case”

4.3.3 総合評価

表 2 に，並列処理環境における各種最適化の有無およびそれらの組合せに対する，スキャン変換時間を示す．表での「Base」は ERT も動的負荷分散も適用しない場合を示している．また，「Base」条件下での各プロセッサのスキャン変換時間を単純平均した値を「DLB(*)」で表している．「DLB(*)」は，ERT を適用しないで動的負荷分散のみを行った場合の理論上の性能限界と考えることができる．今回の実験データでは動的負荷分散のみでは，「Base」に対して 1.3 倍程度の性能改善しか得られない．これに対し，ERT を適用し，ERT 情報の共有ならびに動的負荷分散を適用すると，「Base」に対して視環境 (A) および (B) で各々2.86 倍および 4.32 倍の速度向上が得られていることが分かる．

さらに，視環境 (B) に対して表 1 と表 2 を比較すると，1 台のプロセッサで ERT のみを適用した場合に対して，8 台のプロセッサで ERT のみを適用した場合，約 4 倍の台数効果しか得られなかったのに対して，ERT 情報共有ならびに動的負荷分散を適用することで，約 7.5 倍の台数効果が得られていることが分かる．

5. 関連研究

5.1 Visibility Culling

1 章でも言及したとおり，大規模データの可視化において不可視な物体の描画を省略する Visibility Culling は必要不可欠な技術である．

不透明な面をプリミティブな描画対象としたサーフェイスレンダリング（あるいはポリゴンレンダリング）においては，ボリュームレンダリングとは異なり幾何学的な位置関係のみの情報で Occlusion Culling が可能であるため，従来から非常に多くの研究が行われている[6),19)]．文献 19) では，ある 1 つの画像に対

して，複数の解像度の画像を考え，それぞれに対して Occlusion Map と呼ばれるテーブルを用意し，そのテーブルを階層的に参照していくことで不可視判定を行っている．Occusion Map には，当該 Map が対応づけられた解像度の画像の 1 ピクセルごとに，当該ピクセルに対応する 1 レベル解像度をあげた（詳細にした）画像のピクセル群の中で不透明なピクセルの割合（これを Opacity と呼んでいる）が格納されている．解像度の低い画像に対応づけられた Occlusion Map の情報をもとに保守的に（安全側に）不可視判定☆を行うことで不可視判定のオーバヘッドを軽減し高速化を図るのがこの論文の特徴である．領域単位で保守的な不可視判定を行っているという点では，我々の Conservative Subscreen Termination も同様である．しかしながら我々の提案手法における保守性は，文献 19) の定義に従うと，Opacity が 1 である場合，かつ，セルの最前面が *deepest* 以上の場合のみに不可視と判定するという点にあり，適用方法が異なっている．

規則構造格子ボリュームデータを対象としたボリュームレンダリングにおける Visibility Culling の研究としては論文 7)，11)，12)，20) 等があげられる．これらに共通する基本的な考え方はボリュームデータをいくつかの直方体サブボリュームに分割し，視点に近いサブボリュームによって遮蔽される後方のサブボリュームの処理を中止するものである．視点位置が与えられると，分割された直方体サブボリューム間の視点に対する前後関係が瞬時に求まるため，前から後ろの順に順次処理を行うことで効率的な Visibility Culling が可能である．なお，一般にサブボリュームの数は非構造格子ボリュームデータを構成するセルの数と比べて非常に少ない．そのため，1 回の不可視判定を時間をかけてより正確に行うという手法をとることも可能である．

論文 20) は，規則構造格子ボリュームデータからレイキャスティング法を用いて等値面を高速に生成する手法を提案している．min-max octree と呼ぶデータ構造を用いてサブボリュームの空間的な位置とボクセル値に関する情報を保持し，可視化対象となる等値面と無関係なサブボリュームの処理を省略している．さらに，等値面と交差するサブボリューム内でも等値面とは無関係なピクセルに対するレイキャスティングを省略する Early Scanline Termination と呼ぶ手法を提案している．

文献 7) は大規模構造格子データの並列可視化において，インタラクティブな視点位置や伝達関数の変更に追従することを目的としたものである．すべてのサブボリュームに対して，当該サブボリュームを任意の視点方向から見た際の投影領域内で最も不透明なピクセルの不透明度を求めた結果の表を前処理として作成し，実行時にはこの不透明度の表に基づいて保守的に不可視判定を行うことで高速化する手法を提案している．具体的な不可視判定は厳密な投影領域ではなく，サブスクリーン単位で判定を行うという保守性は我々と同じである☆☆．さらに，文献 7) では，不透明度に関する伝達関数を，複数の基底伝達関数の線形結合で近似計算することで，実行時における伝達関数の変更に柔軟に対応する手法も提案している．この不透明度計算においても不透明度を低く見積もる保守的な近似法を使用している．

文献 11)，12) は並列処理環境下での Visibility Culling の効果を高める目的で，ERT の進捗状況を表で管理して共有する点，部分累積不透明度を活用する点は我々とまったく同じである．しかしながら，文献 11)，12) では pixel 単位の累積透明度を表で管理するのに対して，我々はサブスクリーン単位で terminate 情報（当該 subscreen が terminate 状態であるか否かを示す 1 bit と *deepest* 値）を表で管理している．pixel 単位の情報を共有することで ERT 判定はより正確になるが，1 回の情報共有のために交換されるデータ量が大きくなる．そのため，不可視判定の回数が圧倒的に大きい我々のケースに適用すると ERT によって処理が軽減される効果よりも ERT のための情報共有のオーバヘッドが大きくなり全体として速度低下を招いてしまう．

非構造格子データのボリュームレンダリングでは，規則構造格子データの場合と異なり，セルの前後関係を必ずしも容易に定義できるとは限らない．これは，視点依存の Visibility Cycle[8] の問題に起因する．したがって，Visibility Culling を実現するうえで何らかの工夫が必要となる．

セルの位置関係に依存しない最も簡単な Visibility Culling の手法は，セルを構成する全頂点が完全に透明な場合にセルの描画を省略する手法である[21),22)]．この効果は伝達関数に依存するもののきわめて容易に実装が可能である．

☆ 厳密には，ここで行うのはスクリーン平面上の重複判定であり，奥行き方向の判定は別途行う必要がある．

☆☆ 細かくみると全ピクセルの透明度を考慮する点では我々と異なる．

視線ごとにセルの隣接関係を追跡するレイキャスティング（RC）法を採用し，視線単位の Visibility Culling とも見なせる ERT の適用による高速化を図った研究の例が文献 23) である．この論文では，同一スキャンライン上の隣接する視線間のコヒーレンシを利用するために SPAN-BUFFER と呼ぶ記憶領域を設け，セルとスキャンラインとの交差情報を投機的に計算しておくことで高速化する手法（SBRC 法）がその特徴としてあげられている．しかしながら，実験結果ではスクリーンサイズが小さい場合を除き，ERT を適用した SBRC 法は ERT を適用していないセル投影法と実行時間に有意な差が見られていない．さらにこの論文ではセル投影法では ERT が適用できないと主張している．この主張に対し，セル投影法の実装方式が我々の手法と異なるため直接の比較は不可能であるが，我々はセル投影法に ERT を適用することで有意な高速化を達成できている．

5.2 グラフィクスハードウェアを用いた高速化

本稿ではグラフィクスハードウェアを用いた高速化は考慮していないが，汎用グラフィクスプロセッサ（GPU）が利用可能な環境において非構造格子データの可視化を GPU を用いて高速化する研究が行われている[24)~26)]．文献 24) では，スキャン変換時に必要な補間計算を GPU のテクスチャマッピング機能を用いたテーブル参照☆に置き換えることで処理を高速化する手法が提案されている．文献 25) は，セルを構成する 4 つの面の法線ベクトルやボクセル値の勾配ベクトル等の付加的な情報が与えられるという前提の下にすべてのスキャン変換処理を GPU に行わせることで処理の高速化を図っている．ただし，いずれも事前のソーティングによりセルの前後関係が正しく整列されている（Visibility Cycle が発生しない）ことが適用の前提条件である．これに対して文献 26) の研究は，セルのソーティング自体も GPU に実装することを提案している．この研究では，セルのソーティングを 2 段階に分け，第 1 段階の大まかなソーティングを CPU で行った後，その情報をもとに GPU で 2 段階目の詳細ソーティングと合成処理を行うものである．さらに，文献 26) ではスクリーン単位での不透明判定を行うことで，スクリーン全体が不透明になった場合にそれ以降の処理を打ち切るという Occlusion Culling を行っている．汎用 GPU の高性能化，高機能化の恩恵を受けていずれの手法も高速な描画性能を達成しているが，スキャン変換前に全セルが正確にソーティングされた状況下での逐次処理を前提としているため，初期状態でデータが分散配置されている場合への応用は必ずしも容易ではない．

非構造格子データの可視化に対する別のアプローチとして，非構造格子データを規則構造格子データに変換し，変換後の規則構造格子データに対して GPU 等を用いた高速可視化を行うアプローチがある[15),27)]．前処理としてのデータ変換には時間を要するが，変換後の可視化処理はデータの規則性を利用した種々の高速化手法が使えるという利点がある．しかしながら，変換前のデータと同程度の詳細度を維持すると，一般に変換後のデータ量が爆発的に増加するという問題がある．その結果，変換後のデータ量が GPU の専用メモリに格納できない状態が発生すると十分な高速化は期待できない[3)]．

6. ま と め

本稿では，セル投影型 PVR に ERT を適用し高速化する手法を提案した．動的負荷分散機能を実装したセル投影型 PVR に提案手法を実装し，307,565 の要素からなる非構造格子データを，8 台の PC クラスタ（Pentium4 3 GHz，1 GbE 接続）を用い，300^2 のスクリーン解像度でレンダリングした結果，提案手法を用いない場合に比べて，最良で 4.32 倍の性能向上を達成できることが分かった．

今回の実装では，プログラム自体の最適化がまだ不十分なため，リスト処理の最適化等の今後の改良により計算時間のさらなる短縮が可能であると考えられる．また，CellFast [21)] 等で提案されている種々の最適化手法の適用も今後の課題である．また，提案手法の適用範囲を明確にするためには，サイズや性質の異なる種々のデータを用いて提案手法の効果を評価する必要があり，順次実施していく予定である．一方で，グラフィクスハードウェア（GPU）を利用可能な環境においては，GPU との連携による個々のセル投影処理自体の高速化と本手法との併用も可能であり，今後検討を行う予定である．

さらに，画質に関しては，今回の実装ではスカラ場の勾配を考慮したシェイディングは行っていない．しかしながら，視点位置に依存しない前処理として，各頂点でのスカラ場の勾配を求めることにより[28)]，提案手法に組み込むことは可能であると考える．

謝辞 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．また，多くの貴重なコメ

☆ 可能性のあるすべての組合せに対する補間結果をあらかじめ計算した表を用意しておく．

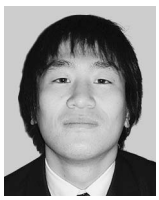
ントをいただいた査読者の方々に大変感謝いたします．なお，本研究の一部は，日本学術振興会科学研究費補助金基盤研究 S（課題番号 16100001），21 世紀 COE プログラム（課題番号 14213201）による．

参 考 文 献

1) 生雲公啓，高山征大，丸山悠樹，津邑公暁，五島正裕，森眞一郎，中島康彦，富田眞治：実時間インタラクティブシミュレーションのための並列ボリュームレンダリング環境，情報処理学会関西支部大会予稿集，pp.121–124 (Nov. 2002).
2) 森眞一郎，中田智史，丸山悠樹，高山征大，生雲公啓，津邑公暁，五島正裕，中島康彦，富田眞治：大規模ボリュームデータの並列可視化環境の構築—専用ハードウェアを用いた実装，先進的計算基盤システムシンポジウム，pp.165–166 (May 2003).
3) 丸山悠樹，中田智史，篠本雄基，高山征大，五島正裕，森眞一郎，中島康彦，富田眞治：汎用グラフィクスカードを用いた並列ボリュームレンダリングシステム，情報処理学会論文誌：コンピューティングシステム，Vol.45，No.SIG11(ACS7)，pp.332–345 (2004).
4) Levoy, M.: Efficient ray tracing of volume data, *ACM Trans. Graphics*, Vol.9, No.3, pp.245–261 (1990).
5) Yagel, R.: Volume viewing algorithms: Survey, *Course Notes for the INTERNATIONAL SPRINGSCHOOL on VISUALIZATION*, Hanspeter Pfister (2000).
6) Durand, F.: *3D Visibility: analytical study and applications*, Ph.D. thesis, Université Joseph Fourier, Grenoble I (July 1999). http://www-imagis.imag.fr
7) Gao, J., Huang, J., Shen, H.-W. and Kohl, J.A.: Visibility culling using plenoptic opacity functions for large volme visualization, *Proc. IEEE Visualization*, pp.341–348 (Oct. 2003).
8) Max, N., Williams, P., Silva, C. and Cook, R.: Volume rendering for curvilinear and unstructured grids, *Proc. Computer Graphics International*, pp.210–215 (2003).
9) 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*), Vol.24, No.5, pp.27–33 (1990).
10) Ma, K.-L. and Crockett, T.W.: A scalable parallel cell-projection volume rendering algorithm for three-dimensional unstructured data, *Proc. IEEE Parallel Rendering Symposium*, pp.95–104 (1997).
11) 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（CPSY），Vol.103, No.249, pp.13–18, 電子情報通信学会 (2003).
12) 松井 学，伊野文彦，荻原兼一：大規模データセット可視化のための効率の良い並列ボリュームレンダリング，情報処理学会論文誌：コンピューティングシステム，Vol.45，No.SIG11(ACS7)，pp.346–355 (2004).
13) 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).
14) Moll, L., Shand, M. and Heirich, A.: Sepia: Scalable 3d compositing using pci pammete, *Symp. on Field-Programmable Custom Computing Machines*, pp.55–118 (1999).
15) Nonaka, J., Kukimoto, N., Sakamoto, N., Hazama, H., Watashiba, Y., Liu, X., Ogata, M., Kanazawa, M. and Koyamada, K.: Hybrid hardware-accelerated image composition for sort-last parallel rendering on graphics clusters with commodity image compositor, *Symp. on Volume Visualization and Graphics*, pp.17–24 (Oct. 2004).
16) 吉村知晋，吉良祐司，森眞一郎，中島康彦，富田眞治：汎用 GPU を用いた大規模可視化クラスタの構築，可視化情報シンポジウム，pp.277–280 (July 2005).
17) 高山征大：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，京都大学大学院情報学研究科修士論文 (2004).
18) Takayama, M., Shinomoto, Y., Goshima, M., Mori, S., Nakashima, Y. and Tomita, S.: Implementaion of cell-projection parallel volume rendering with dynamic load balancing, *Proc. Int'l Conf. on Parallel and Distributed Processing Techniques and Applications* (June 2004).
19) Zhang, H., Manocha, D., Hudson, T. and Hoff III, K.E.: Visibility culling using hierarchical occlusion maps, *Proc. ACM SIGGRAPH*, pp.77–88 (1998).
20) Nebauer, A., Mroz, L., Hauser, H. and Wegenkittl, R.: Cell-based first-hit ray casting, *Proc. EUROGRAPHCS/IEEE TVCG Symp. on Visualization*, pp.77–86 (2002).
21) Wittenbrink, C.M.: Cellfast: Interactive unstractured volume rendering, *HP Labs Paro Alto Technical Report HPL-1999-81* (*R.1*), pp.1–4 (Sep. 1999).
22) Leven, J., Corso, J., Cohen, J. and Kumar, S.: Interactive visualization of unstructured grids using hierarchical 3d textures, *Proc. Volume Visualization and Graphics Symp.*, pp.37–44

(2002).
23) Berk, H., Aykanat, C. and Gudukbay, U.: Direct volume rendering of unstructured grids, *Computers and Graphics*, Vol.27, No.3, pp.387–406 (2003).
24) Ertl, T., Roettfer, S. and Kraus, M.: Hardware-accelerated volume and isosurface rendering based on cell-projection, *Proc. IEEE Visualization*, pp.109–116 (Oct. 2000).
25) Ertl, T., Weiler, M. and Kraus, M.: Hardware-based view-independent cell projection, *Proc. Volume Visualization and Graphics Symposium*, pp.13–21 (Oct. 2002).
26) Callahan, S.P., Ikits, M., Comba, J.L.D. and Silva, C.T.: Hardware-assisted visibility sorting for unstructured volume rendering, *IEEE Trans. Visualization and Computer Graphics*, Vol.11, No.3, pp.285–295 (2005).
27) 渡場康弘，小山田耕二，金澤正憲，ジョルジノナカ，坂本尚久：大規模データ可視化のためのストリーミングベースのボリュームレンダリング手法，可視化情報シンポジウム，pp.387–390 (July 2003).
28) Chen, L., Fujishiro, I. and Nakajima, K.: Parallel performance optimization of large-scale unstructured data visualization for the earth simulator, *Proc. 4th Eurographics Workshop on Parallel Graphics and Visualization*, pp.133–140, Eurographics Association (2002).

(平成 17 年 10 月 4 日受付)
(平成 18 年 2 月 14 日採録)



高山 征大（正会員）

1978 年生．2002 年京都大学工学部情報学科卒業．2004 年京都大学大学院情報学研究科通信情報システム専攻修士課程修了．現在，（株）東芝勤務．

森 眞一郎（正会員）

1963 年生．1987 年熊本大学工学部電子工学科卒業．1989 年九州大学大学院総合理工学研究科情報システム学専攻修士課程修了．1992 年九州大学大学院総合理工学研究科情報システム学専攻博士課程単位取得退学．同年京都大学工学部助手．1995 年同助教授．1998 年同大学大学院情報学研究科助教授．工学博士．並列/分散処理，可視化，計算機アーキテクチャの研究に従事．電子情報通信学会，可視化情報学会，IEEE CS，ACM，EUROGRAPHICS 各会員．

中島 康彦（正会員）

1963 年生．1986 年京都大学工学部情報工学科卒業．1988 年同大学大学院修士課程修了．同年富士通入社．スーパコンピュータ VPP シリーズの VLIW 型 CPU，命令エミュレーション，高速 CMOS 回路設計等に関する研究開発に従事．工学博士．1999 年京都大学総合情報メディアセンター助手．同年同大学大学院経済学研究科助教授．現在に至る．2002 年より（兼）科学技術振興機構さきがけ研究 21（情報基盤と利用環境）．計算機アーキテクチャに興味を持つ．IEEE CS，ACM 各会員．

富田 眞治（フェロー）

1945 年生．1968 年京都大学工学部電子工学科卒業．1973 年同大学大学院博士課程修了．工学博士．同年京都大学工学部情報工学教室助手．1978 年同助教授．1986 年九州大学大学院総合理工学研究科教授，1991 年京都大学工学部教授，1998 年同大学大学院情報学研究科教授，現在に至る．計算機アーキテクチャ，並列処理システム等に興味を持つ．著書『並列コンピュータ工学』(1996)，『コンピュータアーキテクチャ（第 2 版）』(2000) 等．電子情報通信学会 Fellow，IEEE，ACM 各会員．