修 士 論 文

ラジオシティ法の並列計算による画像生成の高速化

指導教官　堀口 進 教授

北陸先端科学技術大学院大学
情報科学研究科情報システム学専攻

阿部 寛之

1997 年 2 月 14 日

目 次

図 目 次

表 目 次

第 1 章

序論

1.1 研究の背景と目的

百聞は一見にしかずの格言どおり、人間にとって画像や映像は各種の情報交換・保存・伝達などにおける最も重要な手段となっており、コンピュータによりそれらの生成を行うのがコンピュータグラフィクス (CG) である。コンピュータの普及とともに、科学的事象の視覚化 (サイエンティフィック・ビジュアライゼーション) や商業・工業デザインの分野、コンピュータアニメーションなど、多くの分野について CG は広く社会に浸透しつつあり、CG を目にしない日はないといってよい。

その中でも 3 次元 CG は、近年の急速なコンピュータの処理能力の向上とともに世の中に採り入れられ、3 次元 CG 生成の様々な技法が次々に発表されている。

3 次元 CG のレンダリング技法の一つであるラジオシティ法は、伝熱工学を応用した大域照明モデルによる画像生成手法であり、室内照明の高品位 CG を生成する際などによく用いられている方法である。ラジオシティ法では、光源からの直接光だけでなく物体間の相互拡散反射も考慮に入れて画像を計算するため、線光源・面光源が作る不均一な影のや、間接照明が多い室内などの表現などに適し、非常に現実感の高い画像を生成できるのが特徴である。

ラジオシティ法ではフォームファクタと呼ばれる、パッチ間で光が到達する割合の計算が重要となる。これは 2 重積分を解くことによって求められるが、パッチ間に障害物がある場合などに積分計算で求めるのは非現実的である。そのため、近似値を求める方法としてヘミキューブ法 [1] が提案された。また、レイトレーシング法を応用した手法も提案さ

れた。

初期のころは、光の平衡状態から得られる連立方程式を解いて解を求めていたが、漸進法 [2] が発表されてからは、これが一般的に用いられるようになっている。漸進法では徐々に解に近づけていくため、途中の計算でもそれなりの結果が得られる点、およびメモリを大幅に節約できる点が有利である。

一般にラジオシティ法では、フォームファクタの計算時間が全計算時間の大部分を占める。この処理には膨大な時間が必要であり、画像生成をリアルタイムに行うことが要求される現在では、その高速化が実用化に向けての大きな問題点になっている。したがってフォームファクタの並列計算による高速化が重要である。

従来の並列化では、ポリゴンやパッチ単位で分割する手法が提案されている [6][18]。さらにパッチを階層的に分割して精度を上げる方法も提案されているが、並列化が難しく、あまり有効なものは存在しない。また、画像生成専用のハードウェアを用いて並列計算する手法 [6] なども提案されている。

本研究の目的は、動的付加分散を考慮に入れたラジオシティ法の並列計算モデルを提案し、その特性を明らかにすることである。基本となるアルゴリズムは、フォームファクタの計算の際にヘミキューブを分割する方法を用いる。また、漸進法において、ラジオシティエネルギーを放射するパッチを複数にして、漸進法そのものも並列化している。

このアルゴリズムを超並列計算機 Parsytec GC-PowerPlus に実装し、アルゴリズムの有効性と、超並列計算機上で実行する際の問題点について比較検討を行う。

1.2 本論文の構成

本論文の構成は次のとおりである。

第 2 章では、超並列計算機のアーキテクチャについて、処理方式の違いや相互結合網に基づいて述べる。また、本研究で用いた超並列計算機 Parsytec GC-PowerPlus128/32 の基本構成、通信方式、処理性能の概略について述べる。

第 3 章では、3 次元画像を生成するに当たって必要な基本概念を述べる。また、レンダリング手法として代表的な、レイトレーシング法とラジオシティ法について説明する。さらに、ラジオシティ法については、ラジオシティ方程式、フォームファクタを求める計算手法について述べる。

第 4 章では、ヘミキューブの分割によるラジオシティ法の並列化手法を提案し、従来行われてきたパッチの分割による並列化手法などとの比較評価を行い、有効性を検討する。

第 5 章では、本研究の結論について述べる。

第 2 章

超並列計算機

2.1 はじめに

並列計算機にはその処理方式、プロセッサ網の違いにより、様々なアーキテクチャが存在している。プロセシングエレメント (以下 PE と呼ぶ) が同時に処理を実行する並列計算機アーキテクチャは、各 PE に対するプログラムの動作、命令の実行形式、PE 間の同期の有無、PE 間の接続方法やその制御法など多くの要素により決定される。現在までの基礎的研究により、並列計算機や超並列計算機に適したアーキテクチャが少しずつ明らかになってきている。

本章では、並列計算機の処理方式とプロセッサ網について説明する。特に、本研究で用いた超並列計算機 Parsytec GC が採用しているメッシュ網について、その性質について述べる。次いで、超並列計算機 Parsytec GC の構成や通信方式、処理性能の概要について説明する。

2.2 処理方式

並列計算機を処理方式の違いにより分類する場合、命令流 (Instruction Stream) とデータ流 (Data Stream) の数に応じて、以下の 4 つに分類する手法が多く用いられている。

- SISD(Single Instruction Stream Single Data Stream)
- SIMD(Single Instruction Stream Multiple Data Stream)

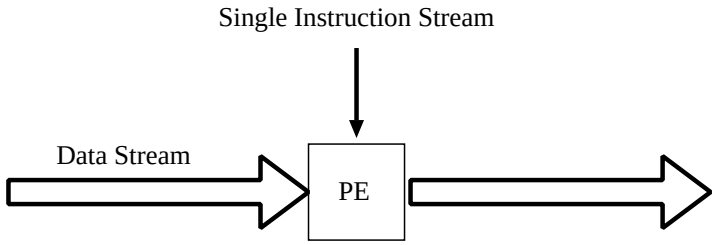


図 2.1: SISD 方式の例

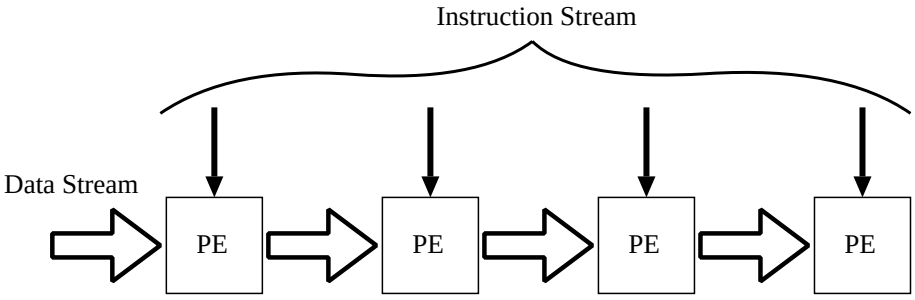


図 2.2: MISD 方式の例

- MISD(Multiple Instruction Stream Single Data Stream)
- MIMD(Multiple Instruction Stream Multiple Data Stream)

SISD 方式は単一命令流単一データ流方式と呼ばれ、図 2.1 に示すように 1 台のプロセッサと記憶装置を持つシリアル計算機、いわゆるフォン・ノイマン型コンピュータである。

MISD 方式は複数命令流単一データ流方式と呼ばれ、図 2.2 のように表すことができる。この方式に当てはまる処理方式としてはパイプライン処理がある。

現在、多くの商用並列計算機の処理方式は、次に示す SIMD、あるいは MIMD 方式である。次の第 2.2.1 節で SIMD 方式について、第 2.2.2 節で MIMD 方式について説明する。

2.2.1 SIMD 方式

一般的に SIMD 方式の並列計算機は、図 2.3 で示すように 1 台のコントロールユニットと、それぞれローカルメモリを持つ多数の PE、そして各 PE 間の通信を行なう相互結

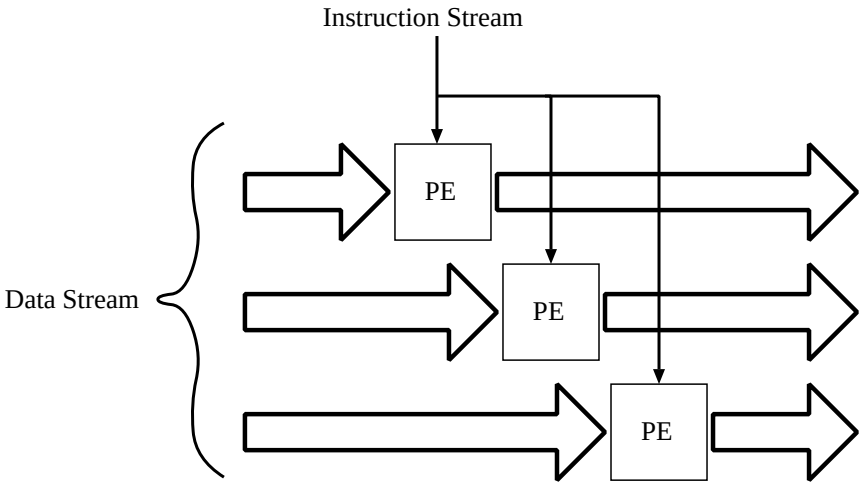


図 2.3: SIMD 方式の例

合網 (Interconnectin Network) からなる。コントロールユニットはプログラムを格納するメモリを持っており、各 PE と結合してインストラクションやデータをブロードキャストし、各 PE は送られたインストラクションを同期して実行する。また、各 PE 内のステータス情報に基づいてインストラクションの実行を抑止することができる。また、結合網自体も SIMD 方式で一斉に制御される場合が多い。

SIMD 方式はインストラクションの流れが通常のノイマン型コンピュータに近いため、並列プログラミングが比較的容易である。また、同様のインストラクションを同期して実行するため、定型的な並列処理の高速化を達成でき、ベクトルや配列といった集合に対する演算を必要とする分野に適している。また、SIMD 方式では、各 PE に順序制御装置が必要でなく、小型化を図ることができ、VLSI 化に適しているなどの利点がある。しかし、コントロールユニットが 1 台であるため、データによって演算内容が各 PE ごとに異なる場合では制御が煩雑になるなど、複雑な処理を行なうことが困難となる問題がある。

2.2.2 MIMD 方式

典型的な MIMD 方式の計算機構成を図 2.4に示す。MIMD 方式は、多数のコントロールユニットと PE の組がそれぞれローカルメモリを持ち、各 PE は相互結合網で接続されている。各 PE は相互結合網を用いて互いに密に通信し、協調しながら独立・並列に動作

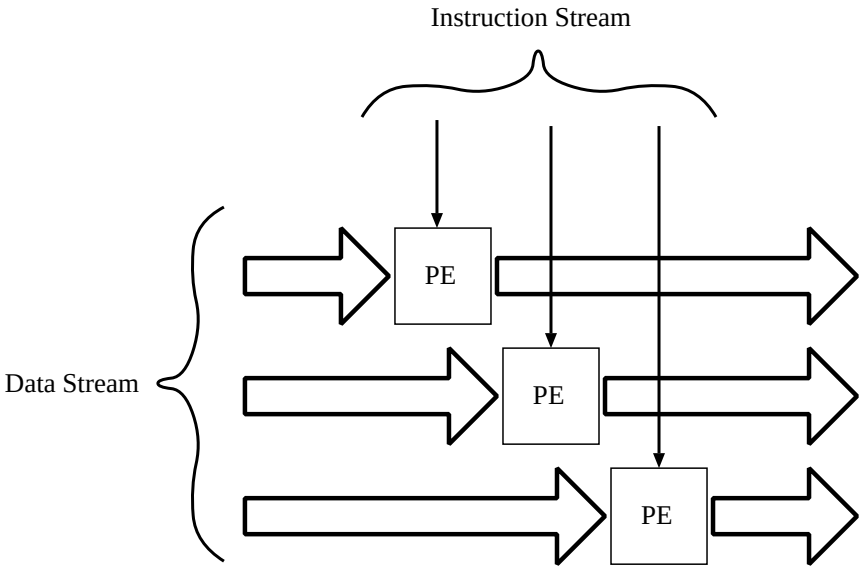


図 2.4: MIMD 方式の例

する。

MIMD 方式は SIMD 方式に比べ自由度が高く、汎用性に優れた処理方式であり、SIMD 方式では困難であった非一様な処理にも対応しやすい。しかし、通信や PE の制御が複雑になる。

MIMD 方式は分類上 SIMD 方式を包含しているので、MIMD 方式ではあるが SIMD 方式にように同一のインストラクションしか実行しないようにもできる。このような考え方に基づいて、SPMD(Single Program Multiple Data Steram) 方式と呼ばれる方式がある。この方式は MIMD 方式に分類されるものの SIMD 方式に極めて近い。SIMD 方式がインストラクションレベルで制御されるのに対して、プログラムレベルで制御を行なうため自由度が高く、SIMD 方式の利点をも併せ持っている。また、MIMD 方式には他にもデータフロー方式などの方式がある。

2.3 結合網

並列計算機の相互結合網の代表的なものとして、次の 3 つがあげられる。

- 格子 (mesh) 結合

表 2.1: 相互結合網のパラメータ

プロセッサ距離	2 つの PE 間で最も中継回数の少ない経路を考え、その中継回数に 1 を加えたものを、この 2 台のプロセッサのプロセッサ距離と呼ぶ。そのうちの最大値を最大プロセッサ距離 D、平均値を平均プロセッサ距離 d とする。
中継量	あるプロセッサにおいて、そのプロセッサを経由する、自分以外のプロセッサどうしを結ぶ通信経路の数。そのうち、最大のものを最大中継量 Rとし、平均値を平均中継量 rとする。
接続数	プロセッサ網を構成するプロセッサ 1 台当たりに直接接続されるプロセッサ数を接続数とし、その平均値を平均接続数 kとする。
放送距離	あるプロセッサ網において、管理プロセッサが放送したデータが全部のプロセッサに到着するまでの時間を、転送回数で表した値を放送距離 Bとする。

- 木 (tree) 結合
- ハイパーキューブ結合

ここで、各結合網の性能を評価するに当たって、必要なパラメータを表 2.1 のように定義する。

2.3.1 格子結合

図 2.5 で示す格子結合網は、実装の容易さから多くの超並列計算機に用いられている。2 次元問題を扱う場合、領域をメッシュに切って部分領域ごとにプロセッサ要素をそのまま割り当てればよいという特徴がある。プロセッサ数が Nの場合、格子結合の各種パラメータは次のようになる。

$$D = 2\sqrt{N} \tag{2.1}$$

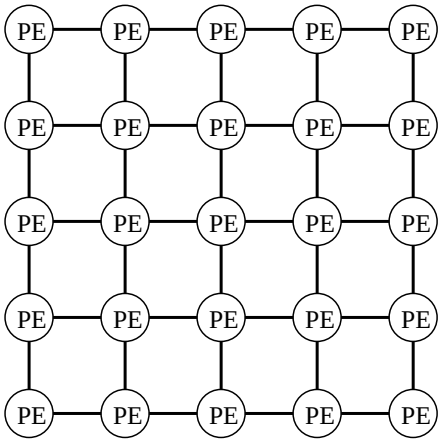


図 2.5: 格子結合

$$d \simeq \frac{2}{3}\sqrt{N} \tag{2.2}$$

$$R \simeq \sqrt{N^3}/2 \tag{2.3}$$

$$r \simeq \sqrt{N^3}/3 \tag{2.4}$$

$$k \simeq 4 \tag{2.5}$$

$$B \simeq \sqrt{N} \tag{2.6}$$

2.3.2 木結合

図 2.6 で示す結合網は、木結合の中でも最も基本的な 2 進木結合である。この結合網はノード間の通信経路が単一であるという特徴がある。探索処理や分割統治法などの木状に処理が進むアルゴリズムを実行するのに向いた方式である。しかし、離れた枝の間では通信経路が長くなり、根の部分にアクセスが集中するため、適用できる問題は限られる。木結合がレベル L の 2 進木結合とすると、プロセッサ数 Nは以下のようになる。

$$N = 2^{L+1} - 1, \text{または } L = \log_2(N+1) - 1 \tag{2.7}$$

また、各種パラメータは次のようになる。

$$D, d \simeq 2\log_2 N \tag{2.8}$$

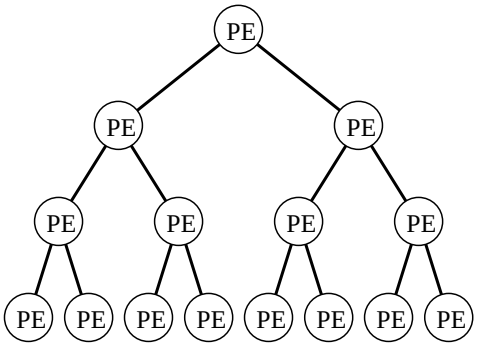


図 2.6: 木結合

$$R \simeq \frac{5}{16}N^2 \tag{2.9}$$

$$r \simeq N \log_2 N \tag{2.10}$$

$$k \simeq 2 \tag{2.11}$$

$$B \simeq 2 \log_2 N \tag{2.12}$$

2.3.3 ハイパーキューブ結合

ハイパーキューブ結合の例を図 2.7 に示す。基本的なハイパーキューブ結合は、2^p個のプロセッサに p ビットの 2 進数で番号を付け、ハミング距離が 1 のプロセッサどうしを接続する方式である。図 2.7 に示したものは、$p = 4$ のハイパーキューブ結合である。ハイパーキューブ結合はプロセッサ距離が小さく、局所的な通信に対しても有効であるが、結線数が多く、平面上にマッピングしにくいという欠点がある。プロセッサ数が N個の場合の、ハイパーキューブ結合の各種パラメータを以下に示す。

$$D = \log_2 N \tag{2.13}$$

$$d = \log_2 N/2 \tag{2.14}$$

$$R = r \simeq N \log_2 N/4 \tag{2.15}$$

$$k = \log_2 N \tag{2.16}$$

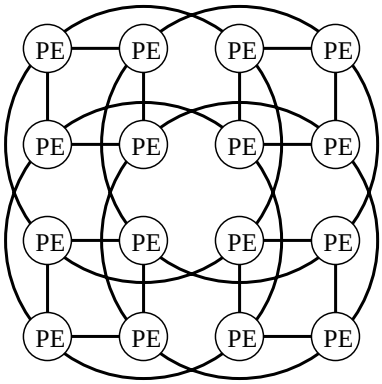


図 2.7: ハイパーキューブ結合

$$B = \log_2 N \tag{2.17}$$

2.4 超並列計算機 Parsytec GC

本研究で用いたドイツ Parsytec 社の GC-PowerPlus128/32 は、米国モトローラ社の RISC プロセッサ PowerPC601 とトランスピュータ通信技術を融合した分散メモリ方式の MIMD 型超並列計算機である。外観を図 2.8 に示す。このシステムの構成図を図 2.9 に示す。また、システムの概要を以下に示す。

GC-PowerPlus128/32 の概要

- PowerPC601 を 128 個搭載
  PowerPC601 のピーク性能
  SPECfp92 ··· 93
  SPECint92 ··· 77
- 物理的には 2 次元格子接続
- 仮想的にハイパーキューブやトーラスなど各種ネットワークが可能
- 総容量 12GB の並列ディスクシステムを装備
- 総メモリ容量 ··· 2GB

図 2.8: Parsytec GC-PowerPlus128/32

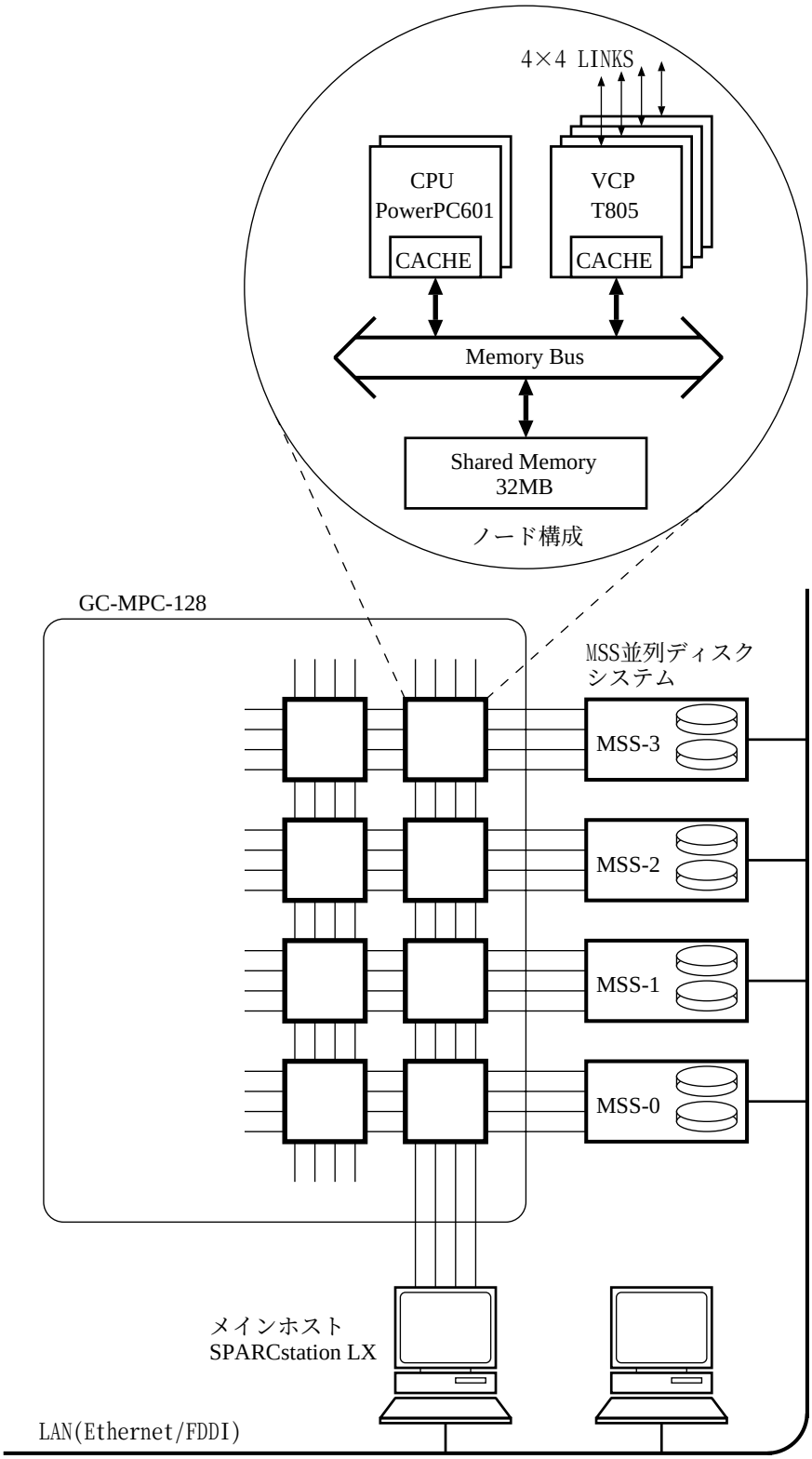


図 2.9: Parsytec GC-PowerPlus128/32 構成図

- 通信性能
  - 通信リンク数 $\cdots$ 16/node
  - 実効通信速度 $\cdots$ 35MB/node
  - メッセージセットアップ時間 $\cdots$ 5μs
  - 最小ネットワークレイテンシ $\cdots$ 60μs
- 最大 I/O バンド幅 $\cdots$ 280MB/sec
- ホストは SPARCstation LX でトランスピュータリンク接続
- O/S は PARIX(Parallel extentions to UNIX)
  - コンパクトなナノカーネル $\cdots$ 約 100KB/node
  - 仮想トポロジ
  - 仮想プロセッサ
  - 同期/非同期通信

GC-MPC-128(並列計算部)

- ノード構成
  - メインプロセッサ $\cdots$ PowerPC601(80MHz) $\times$ 2
  - 通信用プロセッサ $\cdots$ T805(30MHz) $\times$ 4
  - ノードメモリ容量 $\cdots$ 32MB
- ノード数 $\cdots$ 64
- ホスト接続はトランスピュータリンク接続

MSS 並列ディスクシステム

- コントローラ数 $\cdots$ 4
- ディスク台数 $\cdots$ 8
- ディスク総容量 $\cdots$ 12GB
- 接続形態 トランスピュータリンク接続

2.4.1 Computing

GC-MPC-128 では、各ノードに計算用プロセッサとして 60MHz の PowerPC601 を 2 個搭載する。プロセッサ内の communication unit を通じてノード内共有バスに接続されている。

2.4.2 Communication

GC-MPC-128 の各ノードは、communication controller として 30MHz の T805 transputer を 4 個搭載する。各々の T805 は 4 本の通信リンクを持つ。T805 は commuinication bus bridge を通じてノード内共有バスに接続されている。

2.4.3 Memory

GC-MPC-128 では、各ノードに容量 32MB、128 ビット幅の大域共有メモリが用意され、memory controller を通じてノード内共有バスに接続されている。memory controller は 128KB の 8 ビット幅 SRAM を搭載し、高速なメモリアクセスを提供している。

2.5 まとめ

本章では、並列計算機のアーキテクチャについて、処理方式と相互結合網の分類について述べた。

並列処理方式の分類として、SIMD 方式と MIMD 方式を中心に説明した。SIMD 型は、比較的一様な処理を行う場合に適している。また、プロセッサを単純化することができ、最もプロセッサ数を多くとれるアーキテクチャと言える。

MIMD 方式は、SIMD 方式と比較して汎用性が高いかわりに、プロセッサ自体やその制御が複雑になる。

相互結合網の分類としては、格子結合、木結合、ハイパーキューブ結合の 3 つをとりあげ、説明した。

また、本研究で用いる超並列計算機 Parsytec GC-PowerPlus について述べた。GC-PowerPlus は処理方式として MIMD 方式を、相互結合網として物理的には 2 次元格子結合を採用している。しかし、トランスピュータリンク結合により、柔軟な通信が可能で

ある。任意のノード間での通信が可能となっているが、メッセージパッシング型のため通信のオーバーヘッドが処理の高速化に影響している。

第 3 章

3 次元画像生成とラジオシティ法

3.1 はじめに

本章では、まず 3 次元画像の生成に必要な基本的な概念について、簡単に紹介する。次に、代表的なレンダリング手法であるレイトレーシング法とラジオシティ法を取り上げ、それぞれの特徴を説明する。さらに、本論文で扱うラジオシティ法について、ラジオシティ方程式、フォームファクタについて説明した後、問題を解くための 2 種類の計算手法を説明する。

3.2 3 次元画像生成の基礎

3 次元の画像生成とは 3 次元空間中の物体をディスプレイなど 2 次元空間に表示する技法のことである。そのためには、3 次元空間中の位置を特定する座標系を空間に設定しなければならない。また、コンピュータグラフィクスでは最終的に色情報を画像としてディスプレイに出力する。では、コンピュータでは内部で色をどのように表現するのか、コンピュータグラフィクス全体の準備として 3 次元座標系及び色の表現方法について述べる。

3.2.1 座標系

3 次元空間で、その中のある一点と他の全ての点を区別するためには、空間中の全ての点と 1 対 1 の対応がつく量を定義する必要がある。空間中の点の位置を順序を持った実数の組 (座標) で表すとき、空間は座標系を持つ、という。最も代表的な座標系として、原

点と直交する 3 本の直線 (軸) により定義される、直交座標系が挙げられる。X、Y、Z 軸を 3 本の軸とし、空間の 1 点の座標は (x, y, z) で表される。

空間中の任意の座標 (x_1, y_1, z_1) を x 方向に t_x、y方向に t_y、z方向に t_zだけ平行移動したとき、移動先の座標 (x_2, y_2, z_2) は次のようになる。

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \tag{3.1}$$

座標 (x_1, y_1, z_1) を原点を中心に x 方向に s_x倍、y方向に s_y倍、z方向に s_z倍拡大したとき、移動先の座標 (x_2, y_2, z_2) は次のようになる。

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \tag{3.2}$$

座標 (x_1, y_1, z_1) を X 軸、Y 軸、Z 軸周りにそれぞれθ回転したとき、移動先の座標 (x_2, y_2, z_2) は次のようになる。

$$\text{X 軸周り：} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \tag{3.3}$$

$$\text{Y 軸周り：} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \tag{3.4}$$

$$\text{Z 軸周り：} \begin{bmatrix} x_2 \\ y_2 \\ z_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix} \tag{3.5}$$

3.2.2 色

コンピュータでは色を表すのに RGB 表色系が用いられる。これは Red(赤)、Green(緑)、Blue(青) の光の 3 原色の組み合わせ (r, g, b) により表現する方法である。RGB は混ぜ合わせる (r, g, b) の値がそれぞれ大きな値ほど明るくなる加算混合である。RGB それぞれを何ビットで表すかで表現できる色の数が異なる。RGB 各色 8 ビットずつ、合計 24 ビット使用する場合は、各色 256 段階なので、

$$2^8 \times 2^8 \times 2^8 = 256 \times 256 \times 256 = 16777216 \tag{3.6}$$

と約 1677 万色表現することが可能になる。

RGB 表色系以外では、CMY 表色系や HLS 表色系がある。CMY 表色系は Cyan(水色)、Magenta(紫)、Yellow(黄) の色の 3 原色を組み合わせた減算混合であり、HLS 表色系は Hue(色相)、Lightness(明度)、Saturation(彩度) の組で表す方法である。

3.2.3 モデリングとレンダリング

モデリングとレンダリングは 3 次元コンピュータグラフィクスの中核をなすものであり、最も重要なパートである。

モデリングは立体を表現するための作業であり、主なモデルとしては次のようなものが挙げられる。

ワイヤーフレームモデル 立体を輪郭線群またはパッチにより表現する。立体データの入力方法が簡単で、かつ複雑な立体も用意に取り扱えるが、線表示のため高品質な画像表現が行えない。しかしワイヤーフレームモデルは、高速に表示が可能である。

サーフェスモデル 立体を多角形群により表現する。立体を構成する画単位に頂点の座標点列としてコンピュータ内に記憶される。

ソリッド モデル ソリッド モデルは立方体、球、円柱などのプリミティブと名付ける単純な立体を集合演算などの操作を施し、最終的に複雑な立体を構成する方式である。

一方、レンダリングは、モデリングデータを最終的に人が観測できるように可視化する作業であり、多くの手法が提案されている。代表的な手法には次のようなものがある。

隠面・隠線消去法 Z バッファ法やスキャンライン法など。

陰影処理 シェーディング、スムースシェーディングなど。

現実感あるレンダリング テクスチャマッピング、レイトレーシング法 (光線追跡法)、ラジオシティ法。

(モデリング+レンダリング) 処理 フラクタル、ボリュームレンダリングなど。

3.3 レンダリング手法

ここでは代表的なレンダリング手法である、レイトレーシング法とラジオシティ法について、それぞれの特徴を説明する。

3.3.1 レイトレーシング法

レイトレーシング法は、座標系の中に視点とスクリーンを設定し、スクリーン上の各ピクセルの方向から視点に入ってくる光線を逆に追跡し、明るさ、色を求めるという方法である (図 3.1)。したがって、レイトレーシングを行うプログラムの基本的な構成は図 3.2 に示すようになる。ピクセルをどのような順序で処理してもよいということが、この方法の特徴である。光線の追跡の過程で、最初にぶつかったそのピクセルの方向に見える。このため、光線と物体との交点を計算する必要がある。さらに、ぶつかった物体が鏡面反射や屈折を起こす場合には、反射光や屈折光をさらに追跡することにより、画像の中に反射や屈折の効果を表示することができる。

屈折を起こす物体とぶつかった場合には、反射光と屈折光の両方を追跡し、その結果を総合してピクセルに色を与えなければならない。したがって、1 つのピクセルに対する計算であっても、図 3.3 に示すような木構造に従って光の追跡が必要となる。木の上で辺を 1 つ進むごとに、物体との交点計算を行わなければならない。この木は本来、無限の深さ、無限の広がりを持っているが、それでは計算時間が無限となってしまうので、適当なところで追跡を終了させる。追跡終了の条件としては、次のような条件がある。

1. 反射光が散乱してしまうような面にぶつかった場合。この場合には、これ以上の追跡ができないので、ランバードシェーディングなどによって、光の強さを決定する。

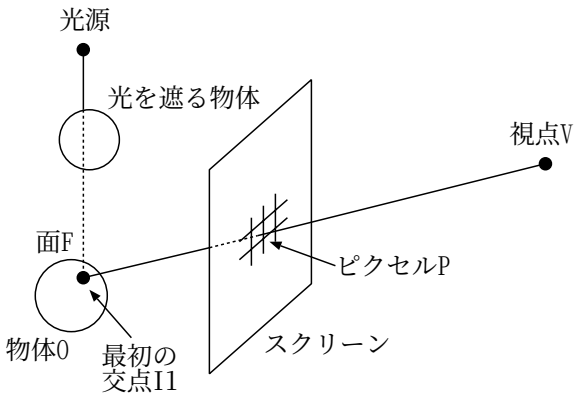


図 3.1: レイトレーシングの原理

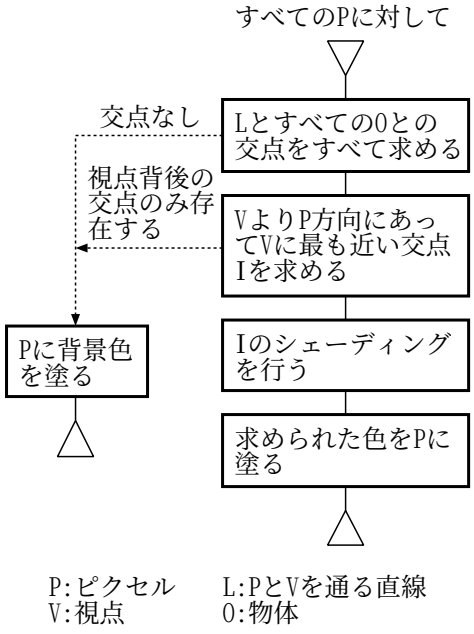


図 3.2: レイトレーシングを行うプログラムの構成

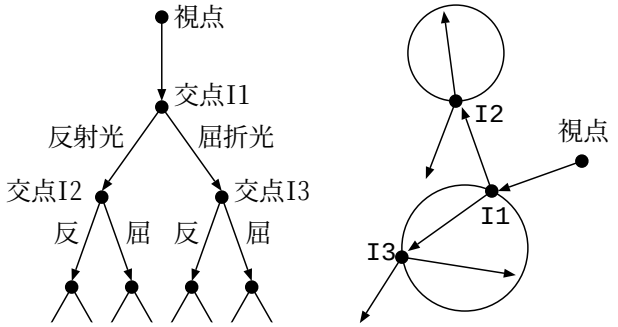


図 3.3: 光の追跡によって構成される木の例

2. 光が物体にぶつからないまま、無限遠に達した場合。このときは背景色によって光の強さや色を決定する。

3. 追跡経路に含まれる反射や屈折の回数が、一定の限度を越えた場合。すなわち、木を十分深いところまで追跡した場合である。反射や透明物体中の透過によって光が減衰するので、反射・屈折の回数や減衰の度合をテストして、最終的な画像に影響が少ないと判断したところで追跡を打ち切る。

追跡する光がある点である面にぶつかったときには、その点が視点から見える。その見えるとわかった点については、光源とその点を結ぶ線分上に他の物体がある稼働かを調べる。何もなければ光が当たっている点であるとし、そうでなければ影の中に入っている点であることがわかる (図 3.1)。ここでも、光線と物体との交点計算が必要になる。

このような計算をすべてピクセルに対して行うのが、レイトレーシングである。なお、影の計算の中で、光源と交点を結ぶ線分上にある物体が透明であったときにはさらにさらに複雑な計算が必要となる。その物体が光源からの光線を屈折させると、交点にどれくらいの光が当たっているのかわからないからである。また、レイトレーシングの問題として、相互拡散反射の扱いがある。レイトレーシングでは、この問題をまったく処理できない。これがレイトレーシングの限界であり、単純なレイトレーシングではこの問題は解決できない。

3.3.2　ラジオシティ法

ラジオシティ法はエネルギー移転の原理に基づく、レンダリング (厳密にはその前処理) 技法である。レイトレーシング法では、拡散反射面の輝度を光源から直接照射される光線のみで決定していたのに対し、ラジオシティ法では、各拡散反射面が放射する光のエネルギーに着目し、閉じた系の中における平衡時の光のエネルギーバランスを求めることにより、拡散反射面の輝度を決定する。したがって、線光源や面光源の作る不均一な影や間接照明が多い室内など、視点に依存しない現象の表現や、緩やかな輝度分布の表現に適している。

3.4　ラジオシティ法

光の方向や物体の配置による影響をまったく考えずに、一様の強さと仮定される光を環境光という。しかし、実際には壁などからの反射を繰り返す相互反射光を考慮する必要があり、これによりリアルさが増す。すなわち、相互反射を考慮することにより次の点において、よりリアルな画像が得られる。

- 影が半影 (ぼやけた影) を伴う。
- 直接光が届かない部分も、相互反射による間接光により照射される。
- 反射面の色が隣接する面に影響する (カラーブリーディングという)。

このように間接光がかもしだす柔らかい雰囲気を表現できるのが特徴である。

相互反射光の計算のことをコンピュータグラフィクスの分野ではラジオシティ法と呼ぶ。この相互反射の計算法としては、物体の構成面をいくつかのパッチに分割して連立方程式を解く方法と、レイトレーシング法を拡張した方法などがある。相互反射の計算では、フォームファクタ (形状要因) が最も重要な要素である。フォームファクタはパッチ間のエネルギーの授受の割合を示し、幾何学的形状のみによって決まるので、光源の高度を変化させても不変である。

また、光源が変化しない場合、相互反射の計算を一度求めておけば、視点が変化した場合でも、再計算しなくてもよい。ただし、鏡面反射成分は視点の位置に依存して変化するから、この場合の処理は複雑となる。

3.4.1 ラジオシティ方程式

まず、光のエネルギーの放射と反射について、すべての放射エネルギーの放射と反射の過程は、理想的な拡散である (入射光がすべての方向に同じ強度で反射する) と仮定する。このとき、ある 1 枚の面から発する光 (ラジオシティ) は、自己放射光 (面自身が発光している) および他の面から入射して拡散反射する光で構成されている。1 つの面から発する光の量を決定するためには、すべての面相互間の幾何学的な関係および各面から発する光の量を記述することが必要となる。そこでまず、あるパッチ i とパッチ jのラジオシティに関する関係を式 (3.7) で表す。

$$B_i A_i = E_i A_i + \rho_i B_j F_{ji} A_j \tag{3.7}$$

ここで、

B_i: パッチ i から発するエネルギー (ラジオシティ) の総量
[エネルギー/単位時間/単位面積]

E_i: パッチ i から自己放射されるエネルギーの総量
[エネルギー/単位時間/単位面積]

A_i: パッチ i の面積

A_j: パッチ jの面積

ρ_i: パッチ i の反射率

F_{ji}: パッチ jから発したエネルギーがパッチ i に到達する割合
(フォームファクタ)

である。ここでフォームファクタとは、あるパッチから他のパッチへ光のエネルギーが到達する率である。フォームファクタの対称性

$$F_{ji} A_j = F_{ij} A_i \tag{3.8}$$

により、式 (3.7) は次のように表すことができる。

$$B_i = E_i + \rho_i B_j F_{ij} \tag{3.9}$$

式 (3.9) は、パッチ i が発する光のエネルギーの総量 (ラジオシティ) は自己放射光と反射光の和に等しいことを示している (図 3.4)。

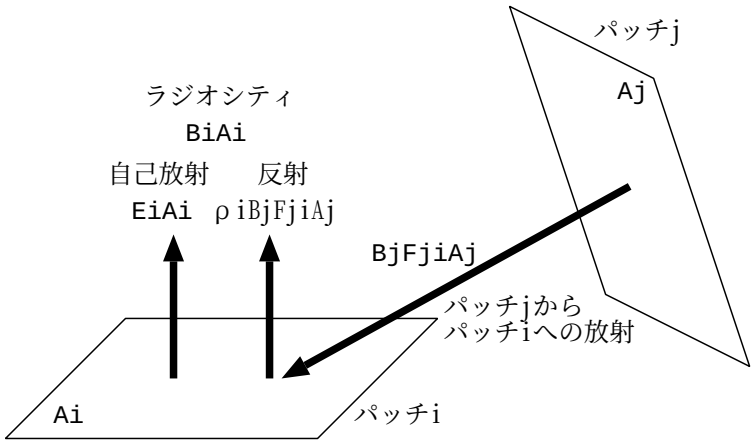


図 3.4: パッチ i のラジオシティ

ラジオシティの解析を行うために、各面を平面パッチ $i(i = 1, \ldots, N)$ に分割する。以下では簡単のために、各パッチは単位面積に分割されたものとする。このとき、式 (3.7) はパッチごとの総和によって、次式により表される。

$$B_i = E_i + \rho_i \sum_{j=1}^{N} B_j F_{ij} \tag{3.10}$$

ここで、Nは総パッチ数である。したがって、系全体としては、

$$\left.\begin{aligned} B_1 &= E_1 + \rho_1 \sum_{j=1}^{N} B_j F_{1j} \\ B_2 &= E_2 + \rho_2 \sum_{j=1}^{N} B_j F_{2j} \\ &\vdots \\ B_N &= E_N + \rho_N \sum_{j=1}^{N} B_j F_{Nj} \end{aligned}\right\} \tag{3.11}$$

の形の連立方程式になる。式 (3.11) を行列の形にすると次式になる。

$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1N} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \cdots & -\rho_2 F_{2N} \\ \vdots & \vdots & & \vdots \\ -\rho_N F_{N1} & -\rho_N F_{N2} & \cdots & 1-\rho_N F_{NN} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} \tag{3.12}$$

これをラジオシティ方程式と呼ぶ。E_iおよびρ_iを設定し、F_{ij}を求め、式 (3.12) を解いて$B_i(i = 1, \ldots, N)$ を求めることにより、各パッチのラジオシティが得られる。式 (3.12) は

$$F_{ii} = 0, \quad \rho_i < 1, \quad \sum_{i=1}^{N} F_{ij} = 1 \tag{3.13}$$

より対角優位行列であるので、ガウスザイデル法と呼ばれる、連立一次方程式の解法によって解を計算することができる。

以上のようにして求めた各パッチのラジオシティにより、各パッチの輝度が決定される。それをもとにして適当な隠面消去、およびレンダリングを行い、画像を生成する。これがラジオシティ法の原理である。

3.4.2 フォームファクタ

式 (3.9) に示したフォームファクタ F_{ij}は、パッチ i の放射するエネルギーがパッチ jに到達する割合を示しており、パッチ相互の幾何学的関係によって決定される。このフォームファクタが求まれば、式 (3.12) を解くことにより各パッチのラジオシティが得られる。ここでは、フォームファクタの求め方を説明する。

いま、パッチ i、jの面積がそれぞれ A_i、A_jであるとする。このとき、パッチ i に含まれる微小領域 dA_iからパッチ jに含まれる微小領域 dA_jへのフォームファクタ $F_{dA_i dA_j}$は次式で与えられる (図 3.5)。

$$F_{dA_i dA_j} = \frac{cos\phi_i \cdot cos\phi_j}{\pi r^2} \tag{3.14}$$

式 (3.14) を領域 A_jについて積分することにより、微小領域 dA_iからパッチ jへのフォームファクタ $F_{dA_i j}$は次式で表される。

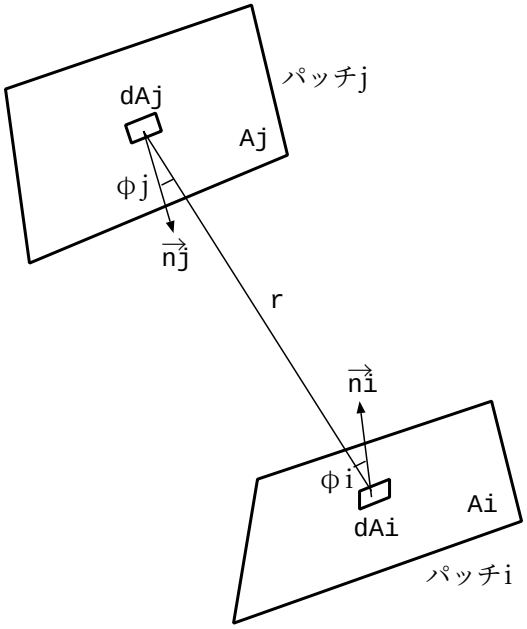
$$F_{dA_i j} = \int_{A_j} \frac{cos\phi_i \cdot cos\phi_j}{\pi r^2} dA_j \tag{3.15}$$

さらに、式 (3.15) を領域 A_iについて積分することにより、パッチ iからパッチ jへのフォームファクタ F_{ij}は領域 A_iにおける平均として次式で表される。

$$F_{ij} = \frac{1}{A_i} \int_{A_j} \int_{A_j} \frac{cos\phi_i \cdot cos\phi_j}{\pi r^2} H(dA_i, dA_j) dA_j dA_i \tag{3.16}$$

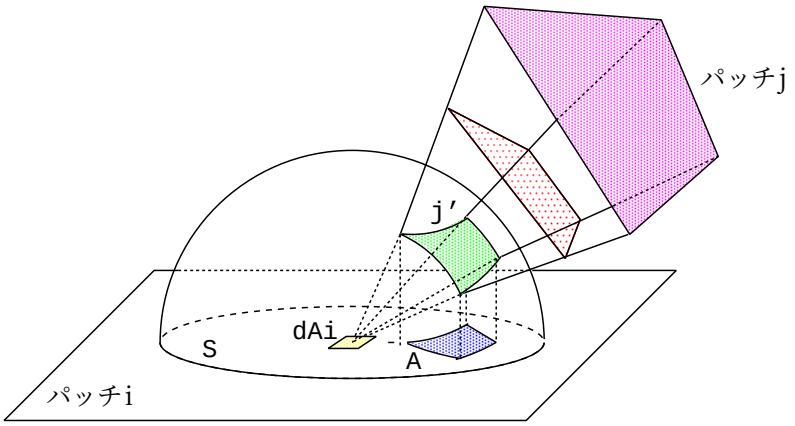
ただし、$H(dA_i, dA_j)$ は、

$$H(dA_i, dA_j) = \begin{cases} 1 & (dA_i \text{と } dA_j \text{の間に遮蔽物体が存在しない場合}) \\ 0 & (dA_i \text{と } dA_j \text{の間に遮蔽物体が存在する場合}) \end{cases} \tag{3.17}$$



$\vec{ni}$:dAiの法線ベクトル　　φi:直線dAi-dAjと$\vec{ni}$との角度
$\vec{nj}$:dAjの法線ベクトル　　φj:直線dAj-dAiと$\vec{nj}$との角度
r:dAiとdAjの距離

図 3.5: パッチ間の関係



j':球に投影されたパッチj　　S:底円の面積
A: j'の正射影

図 3.6: パッチの半球面への投影

で定義される。

フォームファクタは式 (3.16) を用いて解析的に求める必要がある。しかし、積分を含むため非常に多くの計算時間を要し、実用的には計算しにくい。そこで、簡単にフォームファクタを計算する手法が提案された。

フォームファクタの性質として次のことがいえる。パッチ i 上の微小領域 dA_iの周囲に仮想的な半球を設定し、パッチ jを dA_iに向かって投影させる。半球上にパッチ jが投影された領域を j'とすると、

$$F_{dA_i j} = F_{dA_i j'} \tag{3.18}$$

となることが知られている (図 3.6)。すなわち、半球上の同じ領域に投影されるパッチは同じフォームファクタである。

この性質を利用すれば、半球上の領域とのフォームファクタを求めればよい。しかし、半球面上に投影する方法や、半球面上の領域の計算が複雑である。

1985 年に Cohen らは、既存のレンダリング手法を利用して効率よくフォームファクタを算出する、ヘミキューブ法を提案した [1]。この方法では、半球の代わりにパッチ i の周辺に仮想的な半立方体 (ヘミキューブ) を設定する。

まず、図 3.7 に示すように、パッチ i を中心に、X 軸と Y 軸方向に対しては−1 から 1、Z 軸方向に対しては 0 から 1 の大きさのヘミキューブを考える。次に、この表面を適当

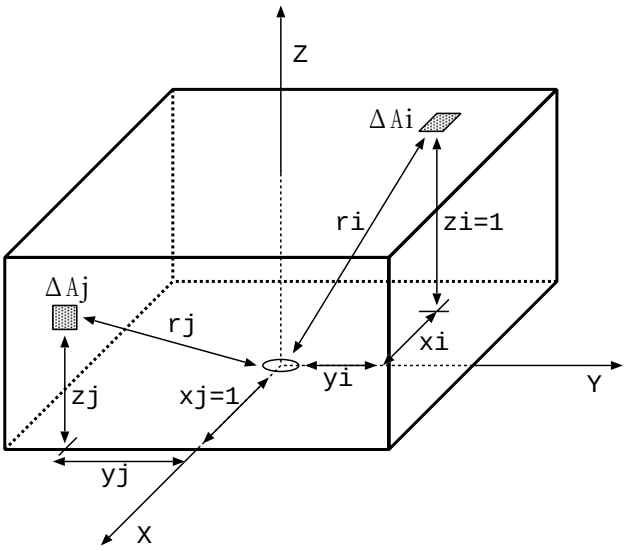


図 3.7: デルタフォームファクタ

な大きさのメッシュに分割する。そして、パッチ i から各メッシュへのフォームファクタ ΔFを次の式を用いて計算する。

上面については、

$$\Delta F = \frac{1}{\pi(x^2+y^2+1)^2}\Delta A_i \tag{3.19}$$

側面については、

$$\Delta F = \frac{z}{\pi(y^2+z^2+1)^2}\Delta A_i \qquad (x=1,-1) \tag{3.20}$$

$$\Delta F = \frac{z}{\pi(x^2+z^2+1)^2}\Delta A_i \qquad (y=1,-1) \tag{3.21}$$

この結果を後の計算のため、ルックアップバッファに入れておく。次に、Z バッファ法などを用いて、パッチ jをヘミキューブに投影し、隠面消去によって距離が最も短い物体のみを残す (図 3.8)。ルックアップバッファを用いて、投影されたメッシュの和

$$F_{ij} = \sum_{q\in j} \Delta F_q \tag{3.22}$$

を計算すると、パッチ i からパッチ jへのフォームファクタ F_{ij}が得られる。

この hemicube 法は、

- Z バッファ法という既存のレンダリング手法を利用

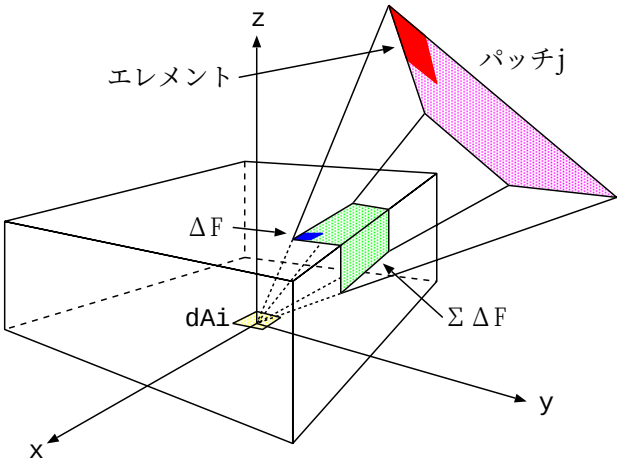


図 3.8: パッチのヘミキューブへの投影

- 半立方体というフォームファクタを計算しやすい形状を利用

しているので、式 (3.16) を直接計算するより効率がよい。

3.5 ラジオシティ法の計算

3.5.1 従来法

従来のヘミキューブを用いたラジオシティアルゴリズムで用いられる計算は、大きく 3 つの部分に分けられる。

1) フォームファクタを計算する

これは、パッチの中心の上に置かれた 5 つの面に他のすべてのパッチを投影し、それぞれの面について Z バッファ隠面消去を適用する。標準的なスキャンコンバージョンと隠面消去のルーチンを使えば、それぞれのヘミキューブの計算量は、離散パッチの数とヘミキューブの解像度に比例する。この結果、全環境内では $O(n^2)$ の計算量が必要となる。

2) ガウスザイデル法を使ったラジオシティ行列を解く

この行列は対角要素が支配的な対角優位行列なので、解は少ない反復で収束し、計算量はパッチ数の 2 乗に比例する。この計算は、それぞれの色要素 (カラーバンド) について行う。

3) 結果を表示する

これには、視線パラメータの選択、隠面消去、ラジオシティ値の補間が含まれる。

ラジオシティ法のほとんどの計算はフォームファクタ計算に費やされる。そこで、計算量を軽減するために、いったんフォームファクタを計算したらその値を記憶しておき、行列計算の際にはこのフォームファクタを繰り返し用いる。原則的にはパッチ数の 2 乗に等しい数のフォームファクタを記憶しておかなければならない。ただ、多くのパッチはお互いに見えないので、係数行列の要素の多くは 0 である。それでも $N \times N$の係数行列は、すぐに日常的に使う記憶容量を越えてしまう。例えば 90 パーセントの行列要素が 0 であるような行列で、1 つのフォームファクタに 4 バイトのメモリを仮定しても、5 万パッチの環境では G バイトのオーダーの記憶容量が必要になる。

漸進的洗練化のレンダリングで重要なのは、完全な解に対抗できるような、十分に使える解を得るまでにかかる処理時間である。今までのラジオシティアルゴリズムでは、すべての環境内のフォームファクタをラジオシティ計算の前に計算する。この計算は $O(n^2)$ の計算量が必要になる。さらに、ラジオシティ方程式でガウスザイデル法を適用するので、すべてのパッチのラジオシティは最初の反復サイクルが完全に終了するまで得られなかった。

3.5.2 漸進法

Cohen らは、パッチ数に比例する程度のメモリ量で、漸近的に画像を生成するラジオシティアルゴリズムを提案した。

パッチ・ラジオシティの同時更新：光の収集と放射

従来法では、パッチ i の自己放射エネルギーおよび他のパッチ jが放射してパッチ i に到達するエネルギーの総和、すなわち、環境から与えられて吸収するエネルギーの総和である。従来手法では、ガウスザイデル法によりパッチ i が収集するラジオシティ値を、パッチ 1 枚ずつ順次求めていた。Cohen らの手法ではこの考え方を逆転してアルゴリズムを構築している。

つまり、パッチは交互に、各々がもつすべての放射エネルギーを瞬時に放射すると仮定した。式 (3.12) の連立方程式の解を得るため用いるガウスザイデル法は、1 度に 1 行ずつ連立方程式を解いていき、求める解に収束させていくものである。第 i 行の方程式を評価

すれば、パッチ i のラジオシティの推定値が出てくる。ただし、この推定値は、他のすべてのパッチのラジオシティのラジオシティの推定値 (その計算ステップの段階での推定値) を基にしている。

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij} \tag{3.23}$$

すなわち、あるシーンで、パッチ i が放射する光量を決めているのは、残りの環境から収集する光である (図 3.9)。式 (3.23) の$\sum$の中の項は、パッチ i のラジオシティに影響を与えるパッチ jの寄与の度合を示している。

$$B_j\text{による } B_i = \rho_i B_j F_{ij} \tag{3.24}$$

この寄与の過程を逆に考えれば、他のすべてのパッチのラジオシティへ、どの程度パッチ i が影響を及ぼしているのか、その寄与の度合もわかる。式 (3.8) を用いると、パッチ i からのパッチ jのラジオシティの影響度は、次のようになる。

$$B_i\text{による } B_j = \rho_j B_i F_{ji} \frac{A_i}{A_j} \tag{3.25}$$

この式は、すべてのパッチ jについて適用できる。したがってパッチ i のラジオシティが環境に及ぼす影響の総和は、次のようになる。

$$\text{すべてのパッチ } j\text{について:} \quad B_i\text{による } B_j = \rho_j B_i F_{ji} \frac{A_i}{A_j} \tag{3.26}$$

パッチ jのラジオシティにこの式を追加していくわけだが、使われるフォームファクタ F_{ij}は依然としてパッチ i のヘミキューブを使って計算できるフォームファクタである。したがって、計算のそれぞれの過程では、ある 1 つのパッチの 1 つのヘミキューブのみを考慮すればよい。そしてその過程で、そのパッチの影響 (そのパッチから環境への放射) を、他のすべてのパッチのラジオシティに追加していく。

このステップを、解が収束するようにパッチ i について数回繰り返される。そのたびにパッチ i のラジオシティの推定値はより正確になっていく。しかし、環境 (i 以外のパッチ) には、以前の B_iの推定値の影響が既に含まれている。この場合、以前と現在の B_iの値の差ΔB_iのみを考慮すればよい。ΔB_iを、未放射ラジオシティと呼ぶ。解法のステップは、次のようになる。

各反復と各パッチ i について：

パッチ i 上のヘミキューブを使ってフォームファクタ F_{ij}を計算する;

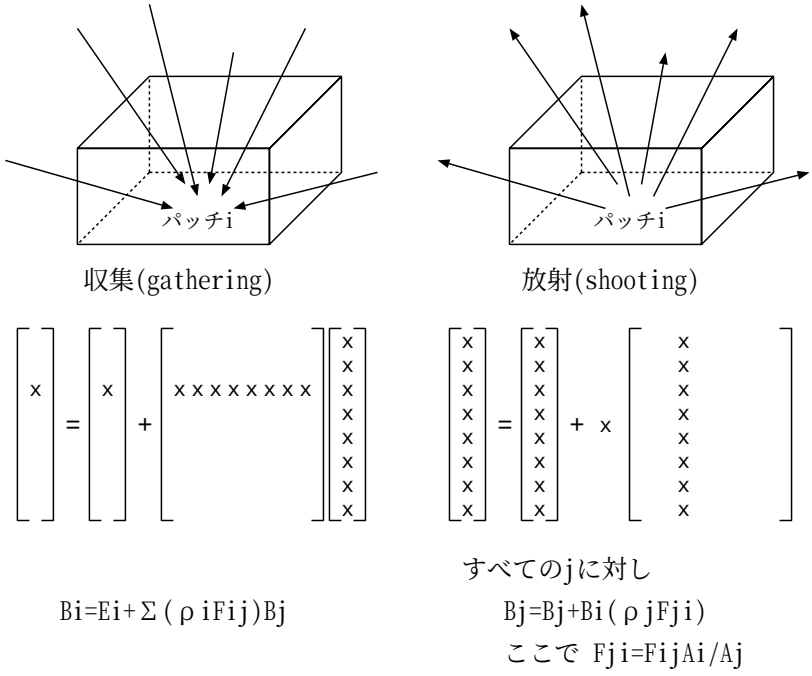


図 3.9: 光の収集と放射

各パッチ jについて：

$\Delta Rad = \rho_j \Delta B_i F_{ij} A_i / A_j;$

$\Delta B_j = \Delta B_j + \Delta Rad;$　/*パッチ jが放射したので誤差を更新する*/

$B_j = B_j + \Delta Rad;$　/*パッチ jのラジオシティを更新する*/

$\Delta B_i = 0;$　/*パッチ i の未放射ラジオシティを 0 にリセットする*/

すべてのラジオシティ、B_iとδB_iは、光源以外を 0 に初期化し、自己放射のあるパッチはその放射値にセットする。上述のステップは適当な誤差以内に解が収束するまで続ける。それぞれの中間ステップで同時に多くのパッチの解が改善される。

ソーティングの適用

上述の光の放出による収束に加えて、できるかぎり早く正確に解が改善されることが望ましい。

パッチ jの最終的なラジオシティB_jは、他のすべてのパッチからの影響の合計である。最も大きい影響を最初に加えるようにすれば、この合計の最終値に最も早く達する。そし

てこれは、最も大きなエネルギーを放射する、すなわち最大の B_iA_iを持つパッチ群の影響である。直観的に言って、最も強い光のエネルギーを放射するパッチは、普通、環境照明に最も強く影響する。このようなパッチは最初に扱うべきである。

そこで、ΔB_iA_iが最大になるようなパッチは、常に放射する。大部分の光源は、この規則に従い自動的に処理される。なぜなら、それ以外のすべてのパッチではラジオシティの初期値は 0 だからである。光源は多くのパッチにとって普通最も重要な照明源なので、光源を最初に処理した後では、多くの環境は既に十分よく明かりに照らされている。この規則に従って処理される次のパッチ群は、その光源から最も多くの光を受け取ったパッチとなる。そして、これが次々に繰り返される。

大きい順にパッチをソーティングし、この順序で処理を進めていくと、光源から光が環境へ伝搬していくのとほぼ同じ順序で進む。これに似たアプローチは、Immel が提案している。視線に無関係な鏡面反射ラジオシティアルゴリズムの効果を増大させるためである。一般に、こうしてパッチをソーティングすると、反復処理の効果には及ばないものの正しい解を与えるようになり、計算量を実質的に削減できる。

3.6 まとめ

本章では、3 次元画像の生成についての基本的な概念を紹介した後、レンダリング手法としてレイトレーシング法とラジオシティ法について述べた。また、ラジオシティ法では従来の手法と漸進法の 2 種類の計算手法について、必要記憶容量と時間計算量や、それぞれの特徴を説明した。

第 4 章

ラジオシティ法の並列化と性能評価

4.1 はじめに

第 3 章でも述べたとおり、ラジオシティ法はリアルな画像が得られる反面、非常に多くの計算時間を要するので、以前から並列計算による高速化の研究がなされている。

ラジオシティ法の並列化のアプローチを大まかな分類に分けると、次のような手法が考えられる。

- ラジオシティ法の (各ステージにおける) 処理内容によって処理を PE に分散させることによって並列化する手法。
- 空間内のパッチを各 PE に割り当てることにより並列計算を行う手法。
- レイトレーシング法を基にした手法で、パッチから放たれる光線を分割し、それぞれを PE に割り当てることにより並列計算を行う手法。
- ヘミキューブをメッシュ状に分割し、各メッシュを PE に割り当てることにより並列計算を行う手法。

従来の研究では、パッチを PE に割り当てて並列化する手法が一般的であった。また、精度を上げるため、パッチを階層分割することによって、物体形状の複雑なところは精度を優先的に計算し、形状の単純なところは計算を減らすといったことも研究されている。しかし、パッチの階層分割による並列化は容易には実現できない。

そこで本研究では、ヘミキューブをメッシュ状に分割し、それを各 PE に分散して並列化する手法を用いている。

本章では、従来のラジオシティの並列化手法を説明し、本研究で用いるヘミキューブの分割による並列化手法を述べた後、超並列計算機 Parsytec GC-PowerPlus への実装と評価について検討する。

4.2 従来の高速化手法

4.2.1 専用ハードウェアを用いた手法

画像生成専用のハードウェアを作成し、それを用いて高速化する研究は、大谷ら [6] や成見ら [14]、Kobayashi ら [19] などによって行われてきた。

大谷らや Kobayashi は、MIMD 型の並列計算機の各ノードに、画像バッファ用メモリとスキャンライン等の処理を行う専用の LSI を搭載したマシンを開発した。

成見らは、ラジオシティ計算専用の Disk Array を開発し、フォームファクタなどをメモリの代わりに高速な Disk Array 上に置く手法を提案した。

いずれも、次に述べるラジオシティアルゴリズムの並列化と併せて用いることにより、高速に処理しようというものである。専用のハードウェアの開発にコストがかかっており、低コストでの開発が望まれる。

4.2.2 パッチの分割と並列化

パッチを PE に割り当てて並列計算する手法は既に多くの研究がなされている。大谷らは専用ハードウェアを用いてパッチの分割による並列化を実現した [6]。この手法の欠点は、PE 間のメッセージ通信が比較的多いので、通信速度が遅い計算機では通信がボトルネックになる [18] ということである。

4.2.3 パッチから放たれる光線を分割する並列化

Lamotte らはレイトレーシングを基にした、パッチから放射される光線を分割し、ラジオシティ法に応用する手法を考案した [7]。この手法はレイトレーシング法ではよく用いられる並列化手法で、光線ごとに PE に割り当てる手法である。

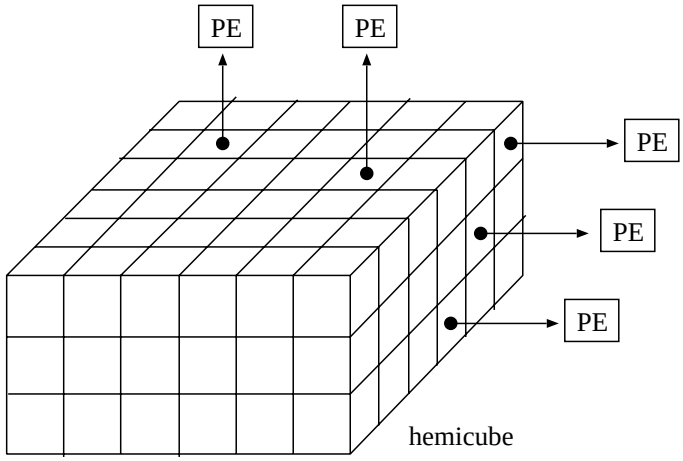


図 4.1: ヘミキューブの分割による並列化

4.3 ヘミキューブの固定分割による並列化

4.3.1 アルゴリズム

本研究で用いた手法は、ヘミキューブを図 4.1 のようにメッシュ状に分割し、各々のセルを PE に割り当てて処理を分散させる方法である。また、漸進法におけるエネルギーの放射では、複数のパッチから同時に放射することによって並列に処理することが特徴である。

この方法によって、フォームファクタの計算とエネルギーの放射の 2 つの段階で並列計算を行っている。本手法の大まかな流れを図 4.3 に示す。

まず、各 PE 内で最大の未放射エネルギーを持つパッチを求め、それをそれぞれの PE がブロードキャストする。次にブロードキャストされたパッチデータから未放射エネルギーの大きいパッチを複数 (本実装では 2 つ) 求める。このパッチのヘミキューブをメッシュ状に分割し、各 PE が担当のデルタフォームファクタを求める。それぞれがデルタフォームファクタをブロードキャストして完全なフォームファクタを得る。各 PE で実際にエネルギーを放射し、ラジオシティの値を更新する。

この作業を未放射エネルギーが少なくなるまで繰り返し、その結果得られたラジオシティ値を Z バッファ法でレンダリングする。

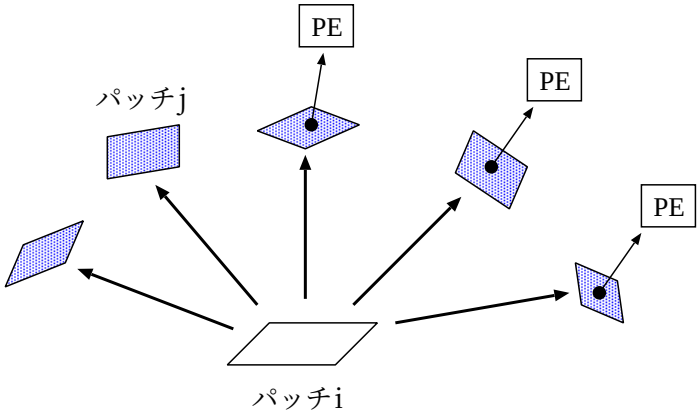


図 4.2: エネルギーの放出の並列化

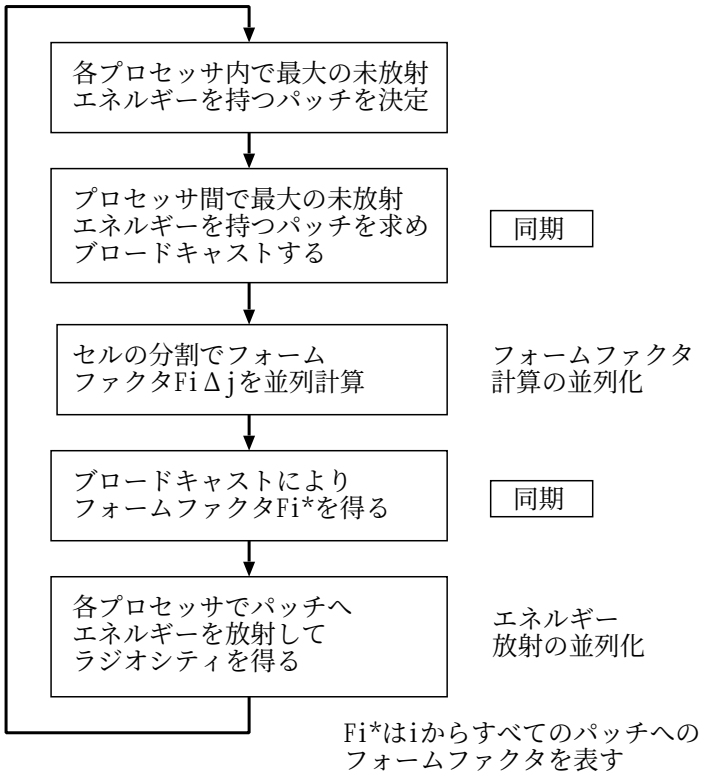


図 4.3: 固定分割法の流れ

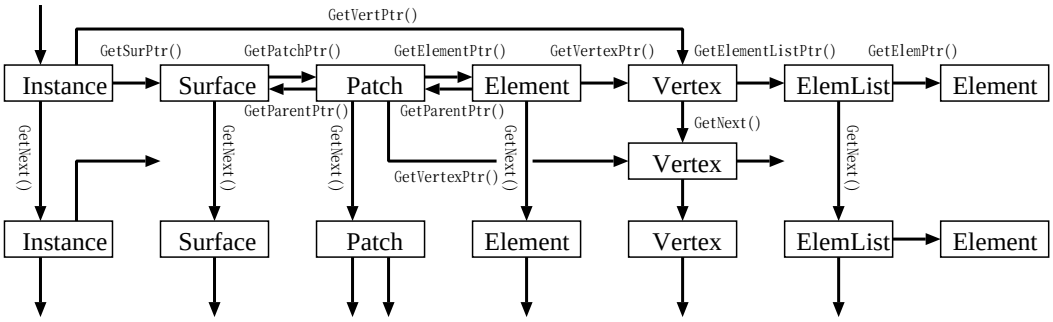


図 4.4: 環境のデータ構造

4.3.2 実装

提案したアルゴリズムを Parsytec GC-PowerPlus128/32 に実装した。プログラミング言語は C++である。GC-PowerPlus では、並列処理のための API は C のライブラリ関数として用意されている。

環境データは図 4.4 のようなリンク構造を持つ構造体として、各 PE のメモリに置かれる。各 PE は 32MB のローカルメモリを持っているので、それぞれのローカルメモリに重複してデータを置く余裕がある。こうすることで PE 間の通信量を極力減らすようにした。入力データファイルは 1 つの環境定義ファイルと 1 つ以上のモジュール定義ファイルから成り、それぞれ図 4.5 、図 4.6 のようなフォーマットで記述しておく。この例は、2 枚の正方形の板が平行に向かい合った、シンプルな空間を示している。

4.3.3 評価

実験には 3 種類のデータを用いた。データファイルの各種パラメータは表 4.1 のとおりである。

この 3 種類のデータから生成された画像を、それぞれ図 4.7 、図 4.8 、図 4.9 に示す。

表 4.2 のように、ループを繰り返すことによりエネルギーが次第に放射されていく。それにしたがって徐々にリアルな画像が得られることがわかる。これらの画像が漸進法によって変化していく過程を図 4.10 、図 4.10 、図 4.10 に示す。

また、放射エネルギー 99.9%に要した処理時間を表 4.3 、図 4.13 に示す。

プロセッサ数が増加するにつれ、計算時間が短縮されていくが、プロセッサ数が一定数

```
WORLD opposing square   /* World name */
COMMENT first square
square.ent              /* Entity finename */
< 1.0 1.0 1.0 >         /* Scaling vector(sx,sy,sz) */
< 0.0 0.0 0.0 >         /* Rotation vector(rx,ry,rz) */
< 0.0 0.0 -0.5 >        /* Translation vector(tx,ty,tz) */
COMMENT second square
square.ent
< 1.0 1.0 1.0 >
< 180.0 0.0 0.0 >
< 0.0 0.0 0.5 >
END_FILE
```

図 4.5: 環境ファイルの形式

```
ENTITY unit square      /* Entity name */
VERTEX
< 0.5 -0.5 0.0 >        /* Vertex vector(x,y,z) */
< 0.5 0.5 0.0 >
< -0.5 0.5 0.0 >
< -0.5 -0.5 0.0 >
END_VERT
SURFACE
[ 1.0 0.0 0.0 ] [ 0.0 0.0 0.0 ] /* Reflectance vector(r,g,b) and  */
                                /* Initail exitance vector(r,g,b) */
END_SURF
PATCH
0 { 0 1 2 3 }           /* Patch id and included vertices */
END_PATCH
ELEMENT
0 { 0 1 2 3 }           /* Element id and included vertices */
END_ELEM
END_ENTITY
```

図 4.6: オブジェクトファイルの形式

表 4.1: 実験に用いた環境データ

	インスタンス数	面数	パッチ数	エレメント数	頂点数
(a) ROOM	9	22	48	197	326
(b) PICROOM	10	27	176	205	343
(c) CORNELL	8	9	112	166	274

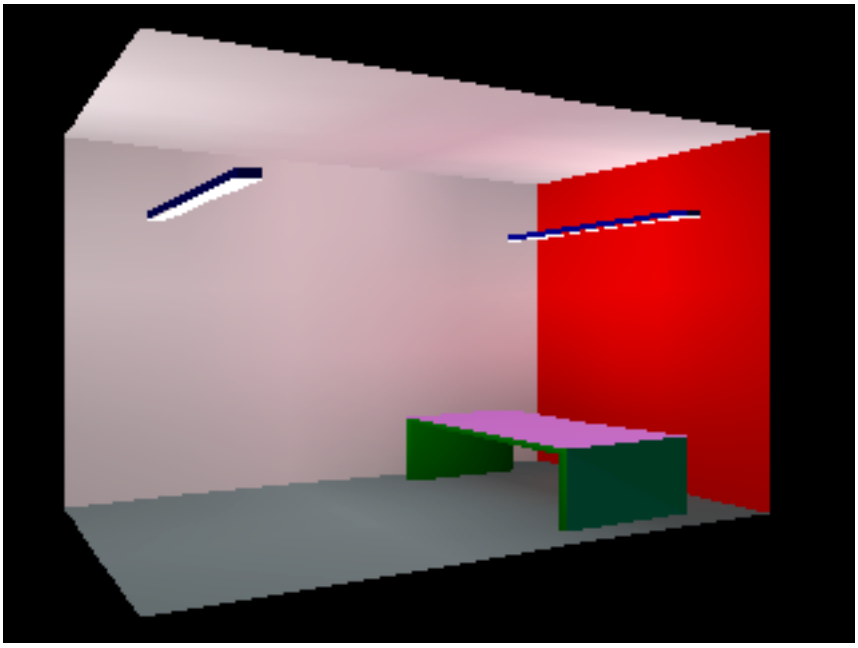


図 4.7: 実験データ画像 (ROOM)

表 4.2: 放射エネルギーと漸進法のループ回数

放射エネルギー (%)	10	20	30	40	50	60	70	80	90	95	99	99.5	99.9
(a) ROOM	4	7	10	12	16	19	25	33	41	44	54	59	73
(b) PICROOM	3	5	7	10	13	30	53	81	123	166	274	307	394
(c) CORNELL	-	-	-	-	1	4	13	30	71	104	183	216	302

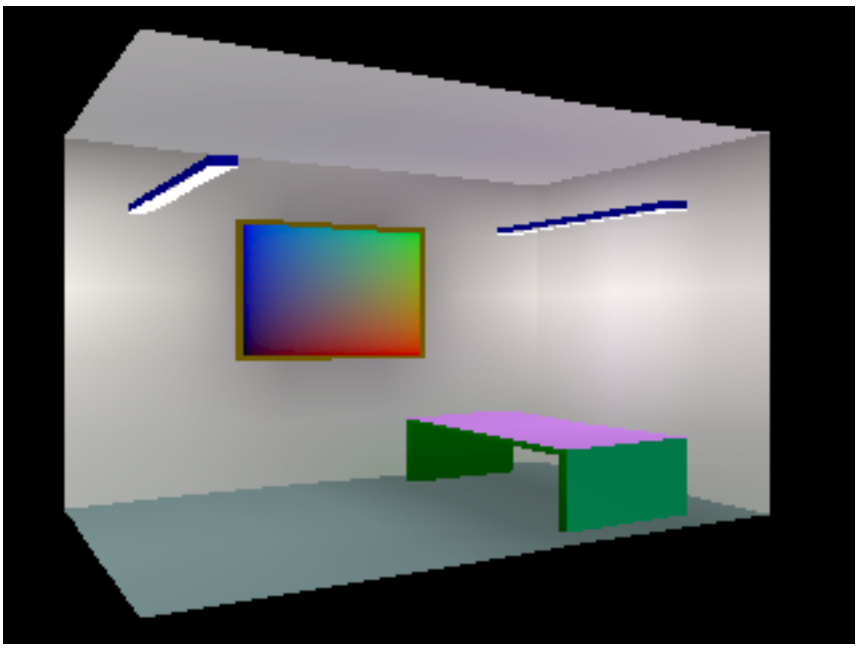


図 4.8: 実験データ画像 (PICROOM)

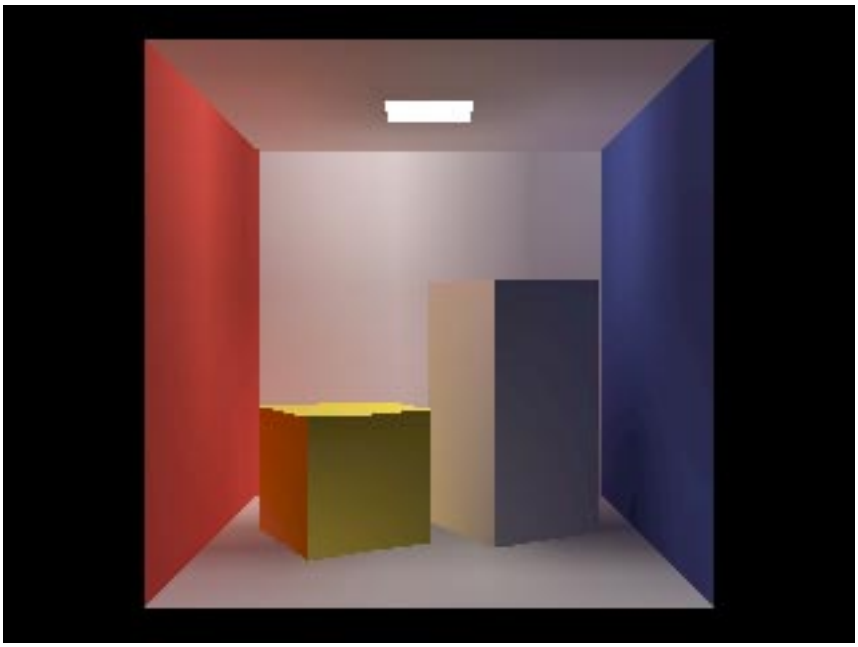


図 4.9: 実験データ画像 (CORNELL)

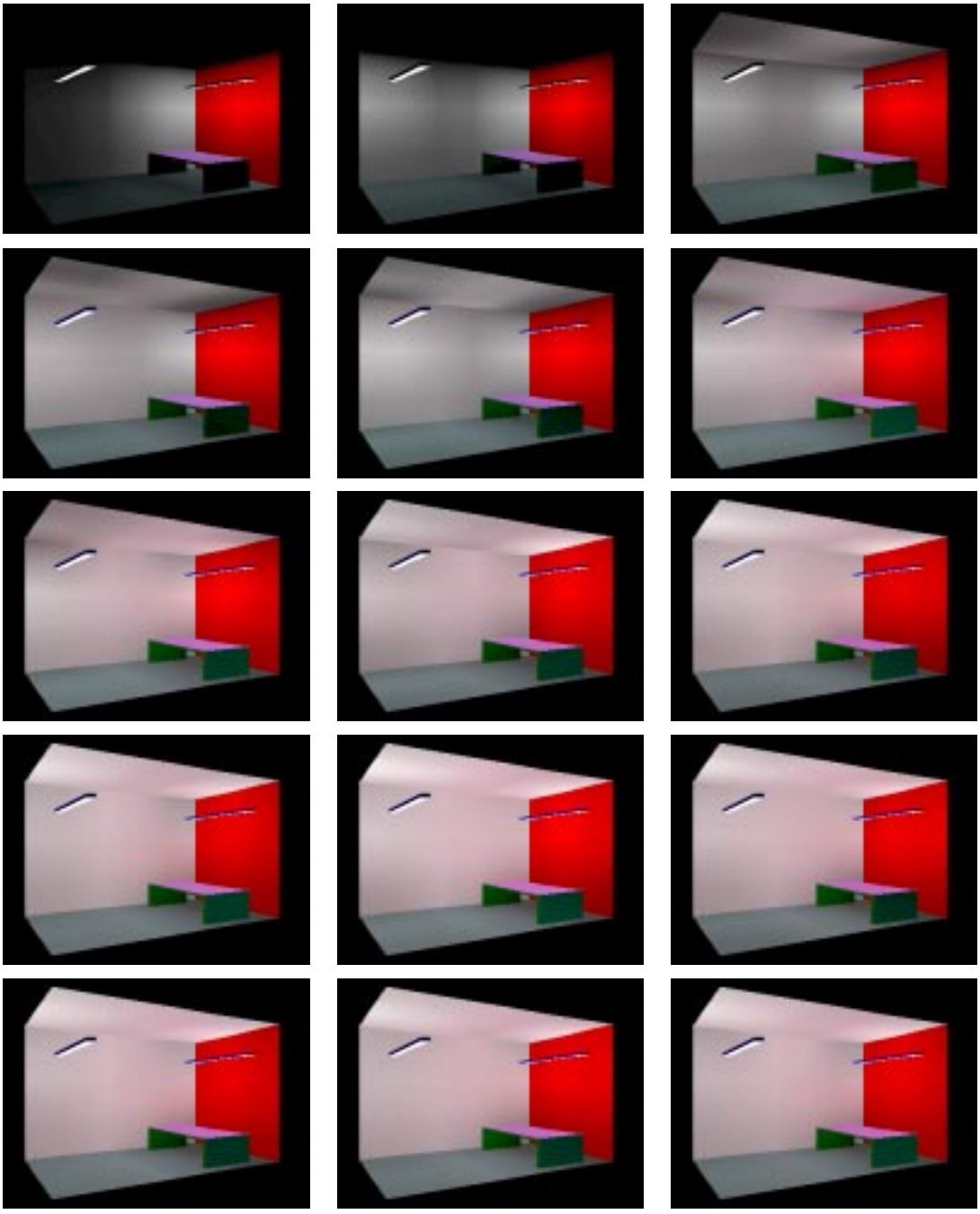


図 4.10: 漸進法による変化 (ROOM)

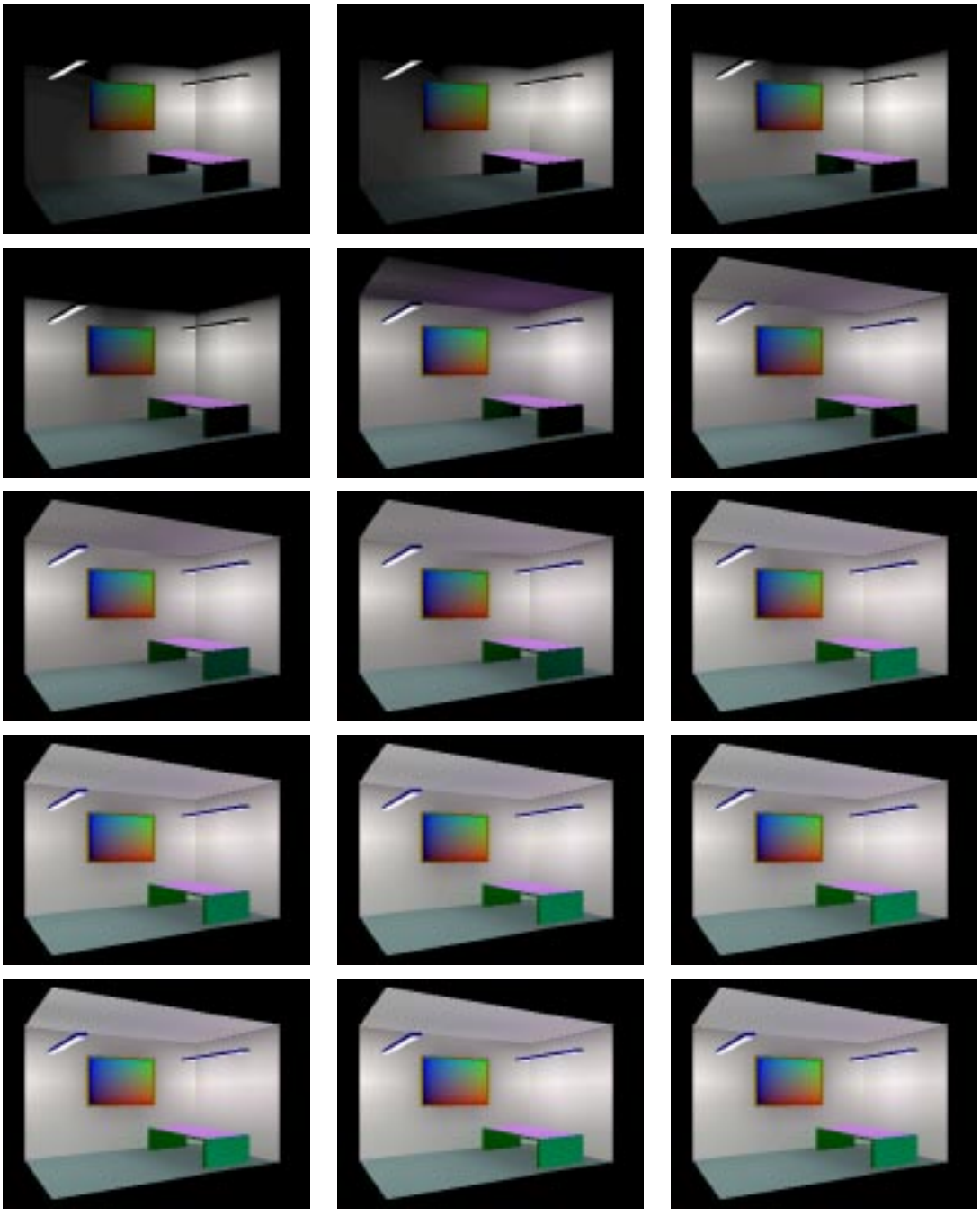


図 4.11: 漸進法による変化 (PICROOM)

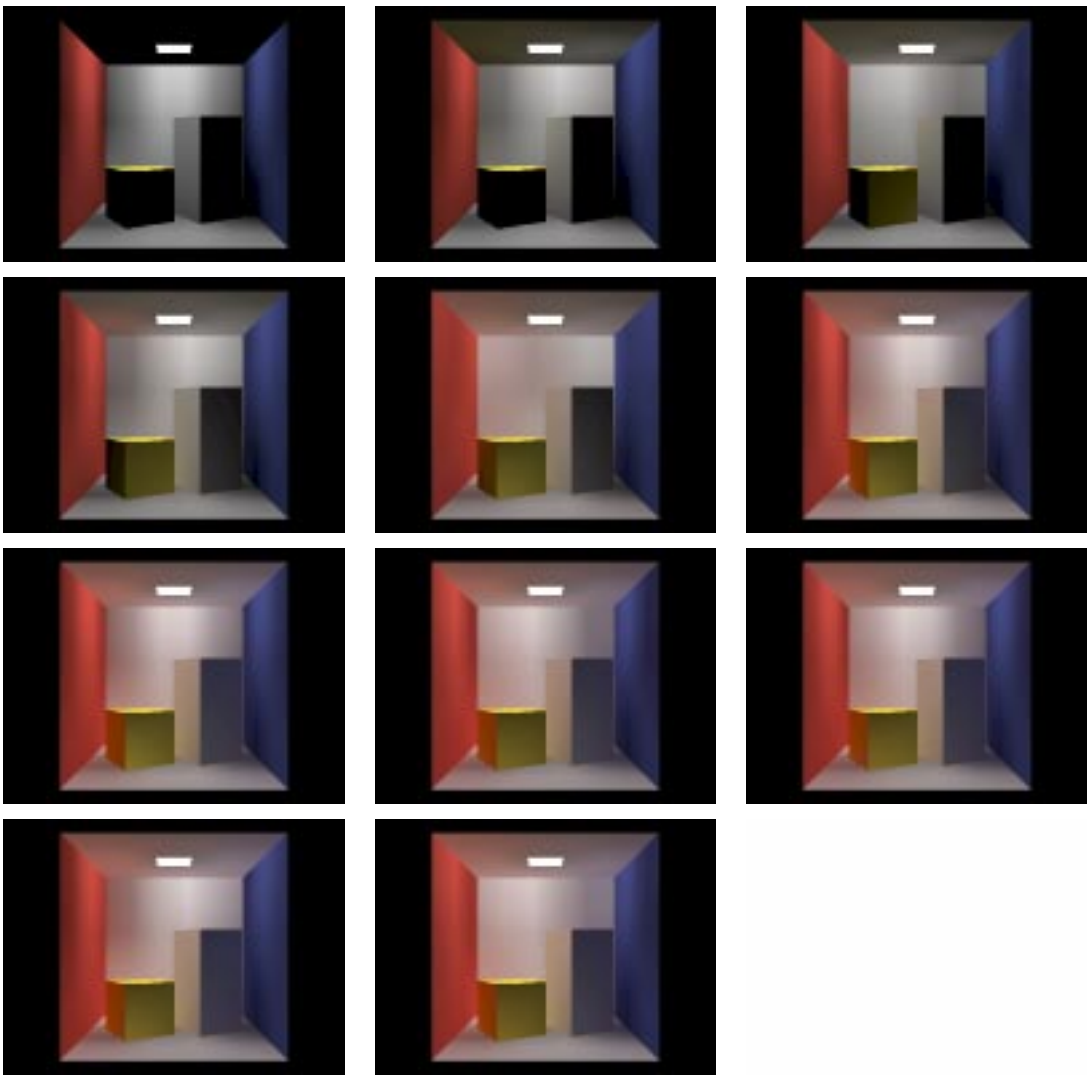


図 4.12: 漸進法による変化 (CORNELL)

表 4.3: ノード数と計算時間

ノード数	1	2	4	8	16	32	64
(a) ROOM(sec)	320	268	221	189	257	384	633
(b) PICROOM(sec)	398	280	232	203	296	429	721
(c) CORNELL(sec)	289	237	186	144	209	336	582

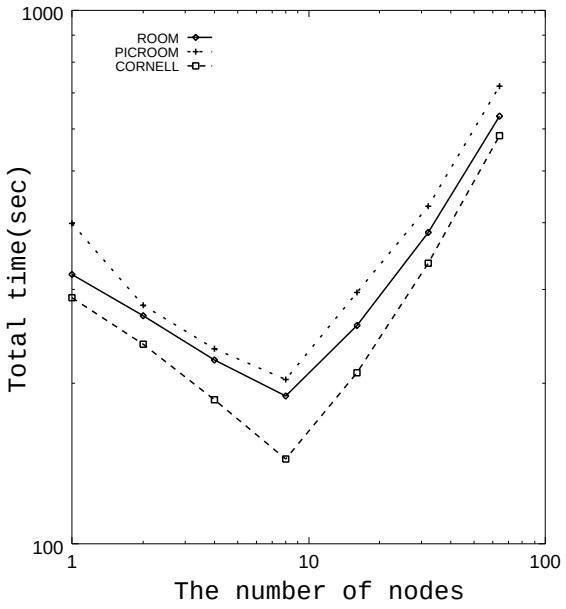


図 4.13: ノード数と計算時間

を越えると再び増加してしまっている。ここで、速度向上比を式 (4.1) のように定義する。

$$\text{速度向上比} = \frac{\text{逐次処理に要する時間}}{\text{並列処理に要する時間}} \tag{4.1}$$

ノード数 1 に対する速度向上比は表 4.4 、図 4.14 のようになる。

また、通信のみに要する時間は表 4.5 、図 4.15 のようになった。ノード数が大きくなるに連れて、全処理時間の大部分が通信にかかっていることが確認できる。この通信時間がボトルネックになって、処理時間が短縮できない原因になっている。

表 4.4: ノード数と速度向上比

ノード数	2	4	8	16	32	64
(a) ROOM(%)	119	145	169	125	83.3	50.6
(b) PICROOM(%)	142	172	196	134	92.8	55.2
(c) CORNELL(%)	122	155	201	138	86.0	49.7

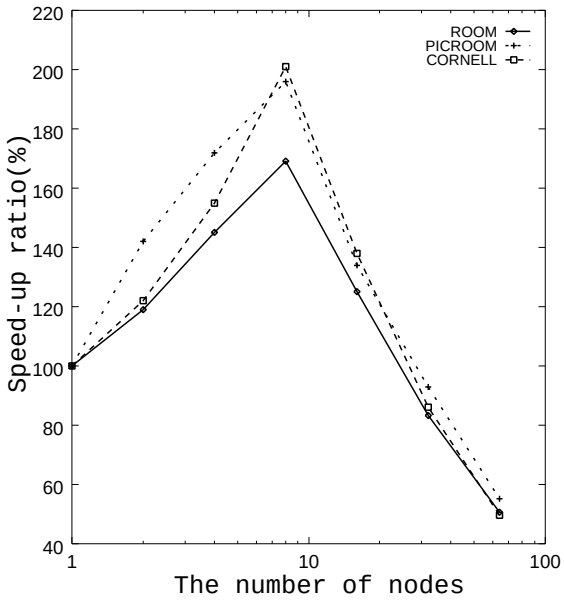


図 4.14: ノード数と速度向上比

表 4.5: 通信にかかった時間

ノード数	2	4	8	16	32	64
(a) ROOM(sec)	25	44	73	180	357	588
(b) PICROOM(sec)	27	48	80	192	389	630
(c) CORNELL(sec)	25	40	66	147	297	442

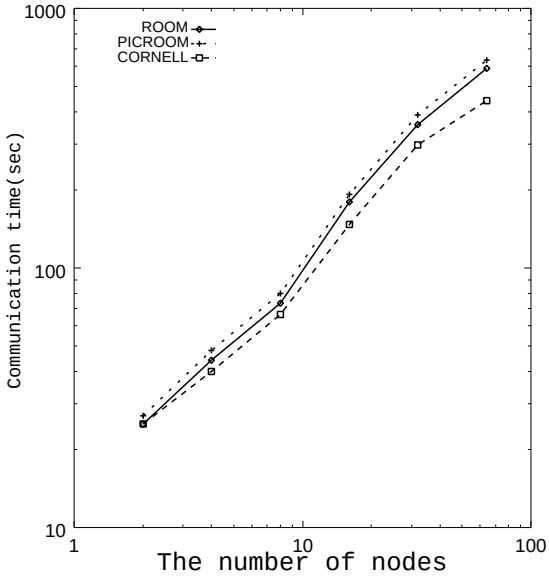


図 4.15: 通信にかかった時間

4.4 通信オーバヘッドを軽減する並列化手法

ノード数が増加するにしたがって処理時間の大部分を通信時間が占めている。これを改善するには次のような方法が考えられる。

- ラジオシティ法のアルゴリズムを改良して同期を減らす
- ブロードキャストを高速化する
- メッセージのサイズを大きくして同期を減らす

これらの方法で通信時間を軽減し、高速化することを考える。

4.4.1 ラジオシティ法のアルゴリズムの改良による同期の軽減

フォームファクタを求めるパッチを最大未放射エネルギーで決定するのではなく、パッチに付けた ID 順に決定する。こうすることによって、ループ中の 1 つ目の同期を解消することができ、高速化が期待できる。このアルゴリズムを図 4.16 に示す。

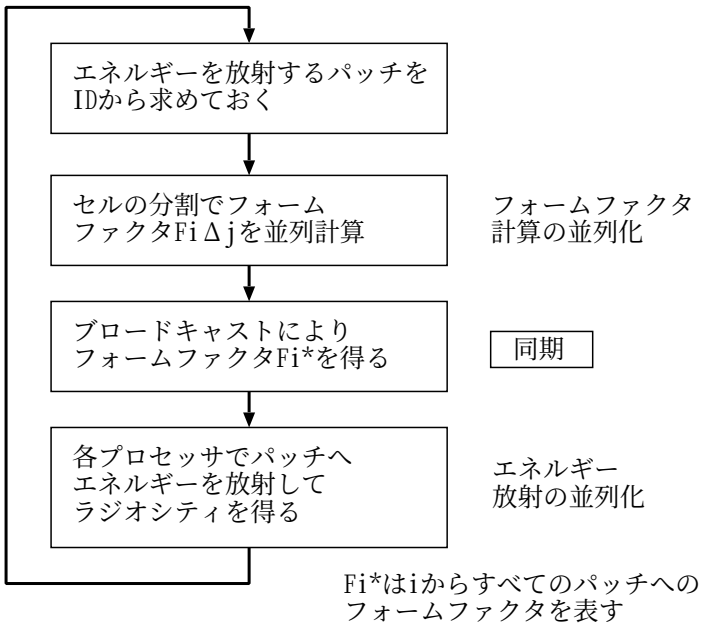


図 4.16: 未放射エネルギーでソートしない手法の流れ

しかし、大きな未放射エネルギーを持つパッチがエネルギーを放射しないので、環境内のエネルギーの収束が極端に遅くなり、事実上停止しないプログラムになってしまった。やはり、環境内のエネルギーのバランスを考えて放射を行うべきである。

そこで、次のような改良策を考えた。この方法では、最初の何回かのループでは未放射エネルギーのソートを行い、全エネルギーに占める未放射エネルギーの割合が減ったらソートをしない。つまり、最初の数回のループでは図 4.3 にしたがって計算を行い、以降は図 4.16 のように計算を行う。こうすることによって、必要なソートは行いながら、無用なブロードキャストを減らすことができるのではないかと考えた。

しかし、この方法もソートを打ち切ってからはエネルギーが収束せず、プログラムが停止しないという状態になった。つまり、ループ中の 2 回の同期は必要であり、これを削ることはできない。

4.4.2 ブロードキャストの高速化

ブロードキャストそのものを高速化することにより、全処理時間に占める通信時間の割合を小さくし、高速化を図る。

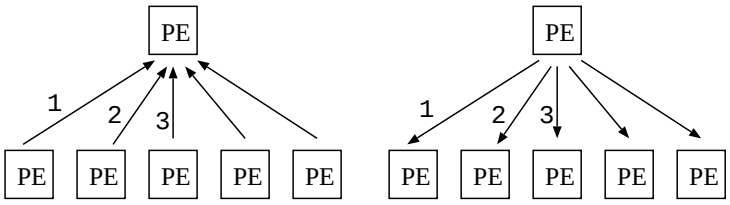


図 4.17: 逐次的通信によるブロードキャスト

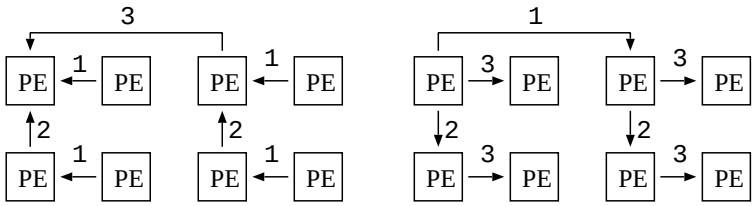


図 4.18: ツリー構造を用いたブロードキャスト

ここまで述べてきた手法の実装では、次のようなアルゴリズムでブロードキャストを行っていた。

すべてのノードからデータを収集する必要がある場合、ある 1 つのノード (root ノード) に対して残りのノードが順にメッセージを送信する。root ノードは同様に残りのすべてのノードに順にメッセージを送信する。このアルゴリズムを図 4.17 に示す。

この方式ではデータの送受信回数は $O(N)$ のオーダーになる。

それに対し、次のようなアルゴリズムでブロードキャストを高速化した。

ツリー構造を考え、末端の葉から順にデータを送信する。そうして最後には root ノードにすべてのデータが到達する。root ノードは今の逆の要領でデータを送り、末端の葉にデータが届くまで繰り返す。このアルゴリズムを図 4.18 に示す。

この方式ではデータの送受信回数は $O(logN)$ のオーダーになる。

この 2 種類の方法を使って、1 度のブロードキャストに要する時間を計測した。逐次的通信によるブロードキャストの結果を表 4.6 と図 4.19 に、ツリー構造を用いたブロードキャストの結果を表 4.7 と図 4.20 に示す。

ノード数が大きい場合、ツリー構造を用いたブロードキャストは、逐次的通信によるブロードキャストに比べて 2 倍以上高速にすることができた。

表 4.6: 逐次的通信によるブロードキャストに要する時間

Nodes	Connect Only	Connect and Transfer		
		1KB	4KB	16KB
2	1.58ms	2.01ms	2.81ms	6.54ms
4	4.79ms	5.61ms	8.41ms	19.6ms
8	21.8ms	23.0ms	27.7ms	48.8ms
16	40.4ms	47.0ms	57.4ms	111ms
32	111ms	137ms	143ms	267ms
64	282ms	343ms	380ms	586ms

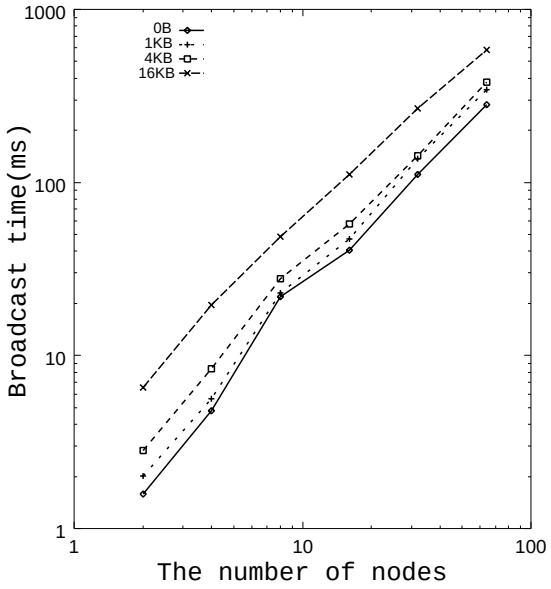


図 4.19: 逐次的通信によるブロードキャストに要する時間

表 4.7: ツリー構造を用いたブロードキャストに要する時間

Nodes	Connect Only	Connect and Transfer		
		1KB	4KB	16KB
2	1.59ms	1.88ms	2.82ms	6.54ms
4	3.89ms	4.65ms	5.93ms	14.6ms
8	19.0ms	19.9ms	22.8ms	33.8ms
16	34.9ms	36.6ms	40.4ms	54.7ms
32	75.3ms	74.7ms	78.1ms	79.1ms
64	103ms	105ms	104ms	137ms

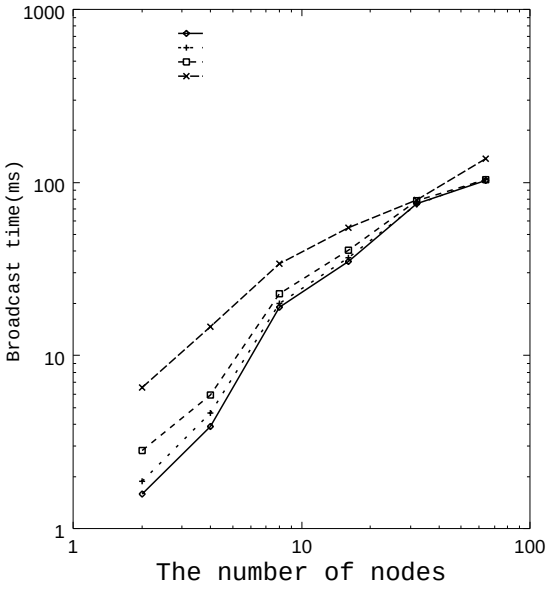


図 4.20: ツリー構造を用いたブロードキャストに要する時間

表 4.8: ノード数と計算時間

ノード数	1	2	4	8	16	32	64
(a) ROOM(sec)	320	267	215	178	227	329	545
(b) PICROOM(sec)	398	279	224	191	264	362	594
(c) CORNELL(sec)	289	236	180	122	177	289	490

このブロードキャストルーチンを用いて、ラジオシティ(放射エネルギー 99.9%) を求めるのに要した処理時間を表 4.8 に示す。また、データごとにブロードキャストを改良する前と後の計算時間を図 4.21 、図 4.22 、図 4.23 に示す。

ブロードキャストアルゴリズムの変更後ではかなりパフォーマンスが改善されていることがわかる。

4.4.3 メッセージのサイズを大きくして同期の回数を低減

この手法による改良は、現段階では実装が完成していない。

ブロードキャストに要する時間からもわかるように、通信スタートアップ時間がかなり大きな値になっている。ノード間にコネクションが 1 度張られてしまえば、比較的大きなデータを送信してもメッセージサイズに比例するほど通信時間が増えないことがわかっている。

1 回あたりのメッセージサイズを大きくして、メッセージの送信回数を減らせば、高速化を見込むことができる。

4.5 ヘミキューブの階層分割による並列化

画像ファイルの精度を上げ、なおかつ計算量を軽減するため、ヘミキューブの階層分割によるフォームファクタの並列計算手法を提案する。各パッチをヘミキューブに投影し、パッチの境界にあたるセルは階層的に 4 つに分割する。漸進法の並列化は第 4.3 節と同様である。ヘミキューブの階層分割による並列化の大まかな流れを図 4.25 に示す。ただし、この手法はブロードキャストが多く発生することが予想できるので、前節で述べた通

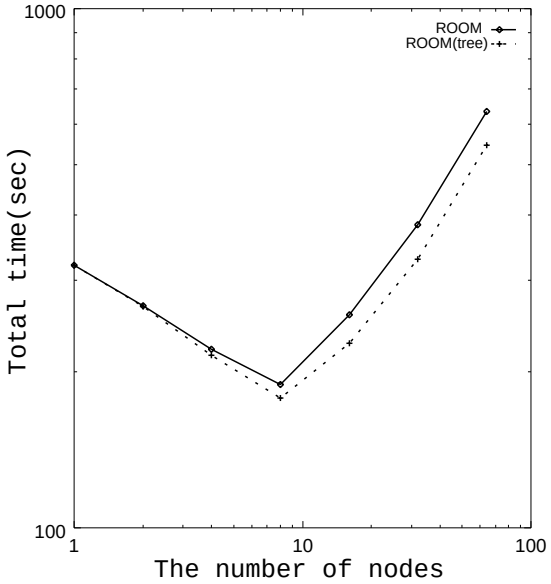


図 4.21: ノード数と計算時間 (ROOM)

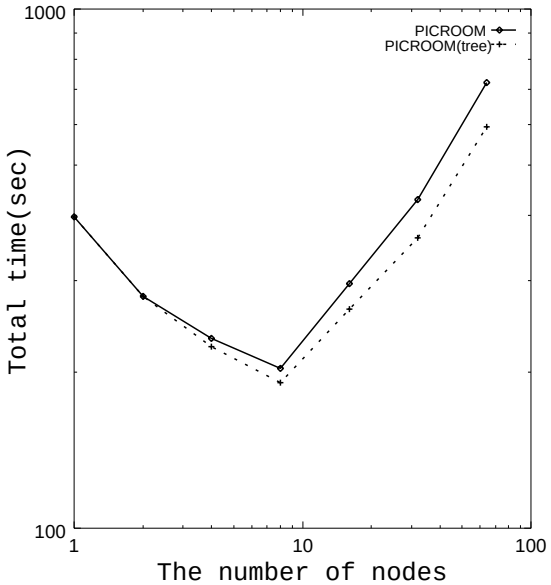


図 4.22: ノード数と計算時間 (PICROOM)

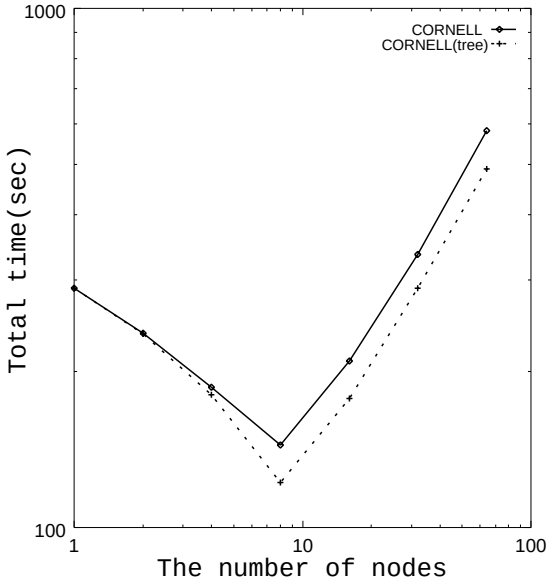


図 4.23: ノード数と計算時間 (CORNELL)

信時間の低減手法を更に工夫して、極力通信を減らす努力が必要がある。

4.6 まとめ

本章では、従来行われてきた並列化手法を紹介した後、ヘミキューブの分割によるフォームファクタの並列化手法を提案した。さらにこの手法を Parsytec GC-PowerPlus に実装してアルゴリズムの有効性を検討した。

プロセッサ数が大きくなると、通信に要する時間が非常に大きくなり、結果として逐次処理よりも速度が劣る結果となった。しかし、ブロードキャストをツリー構造を用いたアルゴリズムに改良することによって、わずかながら処理時間の短縮を図ることができた。

データをブロードキャストするのに、毎回コネクションを張り、データを送り、コネクションを切るという処理を繰り返しているが、コネクションを張るのにもかなりの時間がかかっている。1 回当たりのメッセージサイズを大きくして、メッセージの送信回数を少なくなるようにすば、更に高速化を見込むことができる。

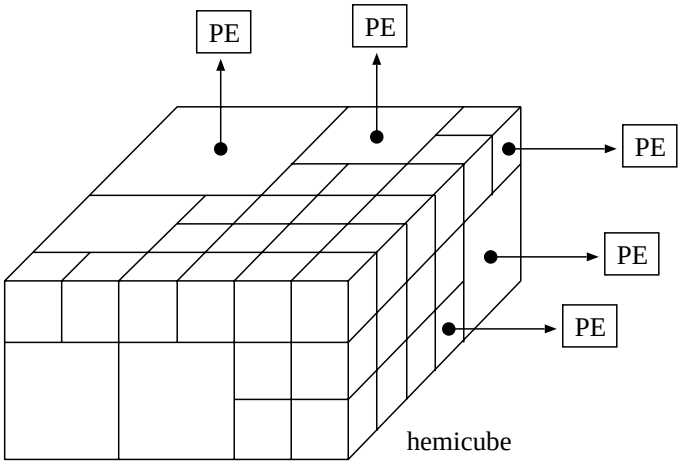


図 4.24: ヘミキューブの階層分割による並列化

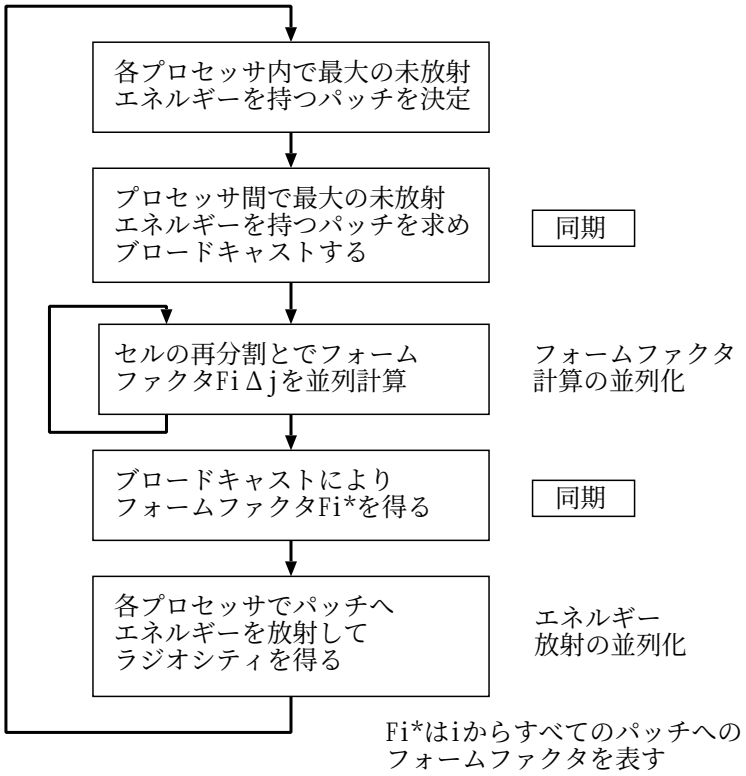


図 4.25: 階層分割法の流れ

第 5 章

結論

5.1 まとめ

コンピュータが普及するにしたがって、CG が様々な分野で使われるようになってきている。中でも 3 次元 CG は、コンピュータの高速化によって急速に社会に浸透しつつある。

レンダリング技法の一手法であるラジオシティ法は、非常に高品質な画像を生成できる手法として注目されている。しかし、処理時間が膨大なため、画像生成をリアルタイムで行うことには無理があった。

本論文では、ラジオシティ法による画像生成の高速化を目的とした超並列ラジオシティ法について提案し、更に分散メモリ型超並列計算機上で実装し、その結果について考察した。以下に本論文の内容を統括する。

第 1 章では、現在用いられている超並列計算機のアーキテクチャについて、処理方式の違いや相互結合網に基づいて述べた。また、本研究で用いた超並列計算機 Parsytec GC-PowerPlus128/32 の基本構成、通信方式、処理性能の概略について述べた。

第 2 章では、3 次元画像を生成するに当たって必要な基本概念を述べた。また、レンダリング手法として代表的な、レイトレーシング法とラジオシティ法について説明した。さらに、ラジオシティ法については、ラジオシティ方程式、フォームファクタを求める計算手法について詳しく述べた。

第 3 章では、ヘミキューブの分割による並列ラジオシティ法を提案し、それを Parsytec GC-PowerPlus に実装した。また、その有効性を検討し、ラジオシティ法のアルゴリズムを変更して同期を減らしたり、ブロードキャストアルゴリズムを変更することで、処理速

度の改善を試みた。また、ヘミキューブの階層分割による並列化手法を提案した。

本研究で行った実験では、プロセッサ数が大規模なシステムになると、プロセッサ間の通信に要する時間が非常に大きくなり、結果として逐次処理よりも速度が劣る結果となった。メッセージを送受信するために必要な通信路の確保に比較的大きな時間がかかっていることがわかった。また、プロセッサ数が大きくなるほど同期に要する時間が非常に大きく影響してくる。

同期そのものを減らすためのラジオシティアルゴリズムの改良や、1 度の同期でのメッセージサイズを大きくして、メッセージの送信回数を少なくなるようにする必要がある。

5.2 今後の課題

今後の課題として、次のことを挙げたい。

- プロセッサ数が大きくなってもスケーラビリティを保つように、プロセッサ間通信方法の改良。具体的にはデータのまとめ送りによる、通信回数の低減。
- ヘミキューブの階層分割によるフォームファクタ計算の並列化の実現、及びパッチの階層分割による手法との各種比較。

謝辞

本研究を進めるにあたり、終始親切なる御指導、御鞭撻を賜わりました堀口進教授に深く感謝致します。

本研究室の阿部亨助教授には、日頃から細部にわたり多大な御指導、御助言を戴き、心より感謝致します。

篠田陽一助教授には、サブテーマで熱心に御指導を戴き、厚く御礼申し上げます。

御教示、御検討を賜わりました日々野靖教授、横田治男助教授に深く感謝致します。

本研究室元助手の沼田一成氏には活発な御指導を戴き、心より感謝致します。

Peralta Rene 客員教授と満保雅浩助手には、並列計算機で御協力を戴き、深く感謝致します。

最後に、日頃より多大な御援助、御協力を戴きました堀口研究室、阿部研究室の皆様に、厚く御礼申し上げます。

参考文献

[1] Michael F. Cohen, Donald P. Greenberg, “The hemi-cube: A radiosity solution for complex environments”, Computer Graphics(SIGGRAPH ’85), Vol. 19, No. 3, pp. 31–40, Aug. 1985.

[2] Michael F. Cohen, Shenchang E. Chen, John R. Wallace, Donald P. Greenberg, “A progressive refinement approach to fast radiosity image generation”, Computer Graphics(SIGGRAPH ’88), Vol. 22, No. 4, pp. 75–84, Aug. 1988.

[3] 天野英晴, 高橋義造 他著, 高橋義造編, “並列処理機構”, 丸善, 1989.

[4] Andrew S. Glassner, 白田耕作訳, “最新 3 次元グラフィックス”, アスキー出版, 1991.

[5] Pat hanrahan, David Salzman, Larry Aupperle, “A rapid hierarchical radiosity algorithm”, Gomputer Graphics(SIGGRAPH ’91), Vol. 25, No. 4, pp. 179–206, July 1991.

[6] 大谷尚毅, 米田泰司, 日高教行, 浅原重夫, 鷺島敬之, “ラジオシティ法の一並列化手法”, 情報処理学会グラフィクスと CAD シンポジウム, pp. 95–104, Sep. 1993.

[7] Wim Lamotte, Frank V. Reeth, Luc Vandeurzen, Eddy Flerackers, “Parallel processing in radiosity calculations”, CGI ’93, pp. 485–496, 1993.

[8] Michael F. Cohen, John R. Wallace, “Radiosity and realistic image synthesis”, Academic Press, 1993.

[9] 阿部毅, 大野義夫, 西田友是, 近藤邦雄, 中嶋正之, “並列グラフィクスアルゴリズムのサーベイ”, 情報処理学会グラフィクスと CAD シンポジウム, pp. 9–16, May 1994.

[10] 池田靖, “並列光線追跡法による動画生成に関する研究”, 北陸先端科学技術大学院大学 修士学位論文, 1994.

[11] 山森一人, “超並列計算機を用いたニューラルネットワークの高速学習法に関する研究”, 北陸先端科学技術大学院大学 修士学位論文, 1994.

[12] Ian Ashdown, “Radiosity: A programmer’s perspective”, John Wiiley & Sons, 1994.

[13] テレビジョン学会編, 中嶋正之監修, “3 次元 CG”, オーム社, 1994.

[14] 成見哲, 牧野淳一郎, 戎崎俊一, 大村皓一, “ラジオシティ計算専用 $DiskArray$システム”, newblock, 情報処理学会グラフィクスと CAD シンポジウム, pp. 19-24, Oct. 1994

[15] 大石進一, 牧野光則, “グラフィックス”, 日本評論社, 1994.

[16] Dirk Lütjens, “Parsytec GC PowerPlus Hardware user guide”, Parsytec Computer, 1994.

[17] Ramona Beyerlein, Helmut Blanke-Bohne el.al., “PARIX Version 1.3.1-PPC Software documentation”, Parsytec Computer, 1995.

[18] 上嶋明, 山崎勝弘, 渡部透, 得丸英勝, “マルチトランスピュータシステム上でのラジオシティ法の並列化”, 情報処理学会並列処理シンポジウム, pp. 19–26, May 1995.

[19] Hiroaki Kobayashi, Hitoshi Yamaguchi, Yoichiro Toh, Tadao Nakamura, “($M\pi^2$): A hierarchical parallel processing system for the multipass rendering method”, IEICE Transactions on Information and Systems, Vol. E-79-D, No. 8, Aug. 1996.

研究業績

[1] 阿部寛之, 阿部亨, 堀口進, “並列ラジオシティ法による画像生成の高速化”, 電気関係学会北陸支部連合大会, E-28, pp. 288, Oct. 1996.