ヘミキューブの分割によるラジオシティ法の並列化

上嶋 明　　山崎 勝弘　　渡部 透　　得丸 英勝
立命館大学理工学部

ラジオシティ法では現実感の高い画像を生成できるが，フォームファクタを求めるために非常に大きな計算量が必要である．本研究では，この計算に用いるヘミキューブの処理を各プロセッサに分割して行わせる方法で並列化した．この際に生じる各プロセッサ間の処理時間の差を小さくするため動的負荷分散法を用いた．並列計算機トランスピュータ64台のシステム上で実験した結果，1台の場合に比較して約20～25倍の速度向上が得られた．

1　はじめに

ラジオシティ法は，物体表面を構成するポリゴンを細かく分割し，その一つ一つについて光の出入りを計算して輝度を求める方法である．ラジオシティ法の特徴は，光の相互反射を取り扱えるため，非常に現実感の高い画像を生成できることである[1][2]．

ラジオシティ法は計算量が非常に多いが，並列処理を行うことで計算時間を短縮できると考えられる[3]．そこで本研究では，ラジオシティ法による画像生成をトランスピュータシステム(25MHz T805×64台)上で並列処理する方法と，その効果について述べる．

すでに提案されているラジオシティ法の並列化の手法では，たとえば文献[4]で示されるように，ポリゴン，パッチやエレメント単位で分割を行うのが一般的である．本研究での並列化の手法は，フォームファクタ計算の際にヘミキューブを分割して各プロセッサに処理を割り当てる方法を用いる．この方法は，各プロセッサ間の通信量が少なくて良いため，トランスピュータシステムのように通信速度が比較的遅い分散メモリ型の並列計算機で有効であると考えられる．ただし，単純にヘミキューブを各プロセッサに等分割したのでは，ポリゴンの配置の偏りが原因となり各プロセッサ間の処理時間に差が生じる．そのため，各プロセッサの処理時間が均等になるように割り当てる必要がある．本研究では各プロセッサへの割り当ての際に単純に処理を各プロセッサに等分割する静的負荷分散法と，実行時に割り当てを行う動的負荷分散法を用いて実験を行い，それぞれの速度向上を示す．

2　トランスピュータシステム

本実験で用いるトランスピュータT805は32ビットのCMOSマイクロプロセッサで，動作クロック周波数は25MHzである．メモリは各トランスピュータに1Mバイト備えている．外部とは，10Mbpsの4本のINMOSシリアルリンクを通じて通信を行う．結合網として，4×4，4×8，8×8の3種類のトーラス，及

Parallelization of the Radiosity Method by Dividing Hemi-cubes
Akira Uejima, Katsuhiro Yamazaki, Tohru Watanabe, and Hidekatsu Tokumaru
Faculty of Science and Engineering, Ritsumeikan University

び2台ずつの組み合わせによる2^5のハイパーキューブを切り替えて使用できる．

本研究では言語としてParallel Cを用いた．Parallel Cは一般のC言語に対して，プロセッサ間の通信やマルチスレッド処理を行うための関数などが拡張されている．マルチスレッド処理を行うことで，計算と通信を並列に行うことができる．

3　ラジオシティ法

3.1　計算手順

ラジオシティ法による画像生成の手順は，以下のようになる[1]．

1. 物体をポリゴン（理想的な拡散反射面）で定義し，適当な大きさのパッチに分割する．
2. 光のエネルギーが平衡状態に達しているとして，全てのパッチについて光の入射と反射に注目すると，以下の式が成立する．

$$B_i A_i = E_i A_i + \rho_i \sum_{j=1}^{n} B_j F_{ji} A_j \qquad (1)$$

   ここで，

   B_i ： パッチiのラジオシティ・エネルギー
   E_i ： パッチiの自発光強度
   A_i ： パッチiの面積
   A_j ： パッチjの面積
   F_{ji} ： パッチjからパッチiへのフォームファクタ
   ρ_i ： パッチiの反射率
   n ： パッチ数

   フォームファクタの対称性により，式(1)は次のように表せる．

$$B_i = E_i + \rho_i \sum_{j=1}^{n} B_j F_{ij} \qquad (2)$$

3. 式(2)より次の連立方程式が立てられ，これを解けば全てのパッチのラジオシティが求められる．

$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \dots & -\rho_1 F_{1n} \\ -\rho_2 F_{21} & 1-\rho_2 F_{22} & \dots & -\rho_2 F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_n F_{n1} & -\rho_n F_{n2} & \dots & 1-\rho_n F_{nn} \end{bmatrix} \cdot \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_n \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix} \qquad (3)$$

4. 各パッチの頂点の輝度を求め，線形補間を用いてレンダリングする．

3.2　フォームファクタ

フォームファクタは，あるパッチから他のパッチへ光が到達する割合である．

2つのパッチi, jが図1のような位置関係にあるとき，フォームファクタは，以下のように表される．

$$F_{ij} = \frac{1}{A_i} \iint \frac{\cos\phi_i \cos\phi_j}{\pi r^2} dA_i dA_j \qquad (4)$$

2つのパッチの間に障害物が存在するような複雑な位置関係の時に，この式によりフォームファクタを正確に計算するのは非常に大きな時間を要して非現実的である．そこで，高速に近似値を求める方法が研究された．その1つに，次項で述べるヘミキューブ法がある．

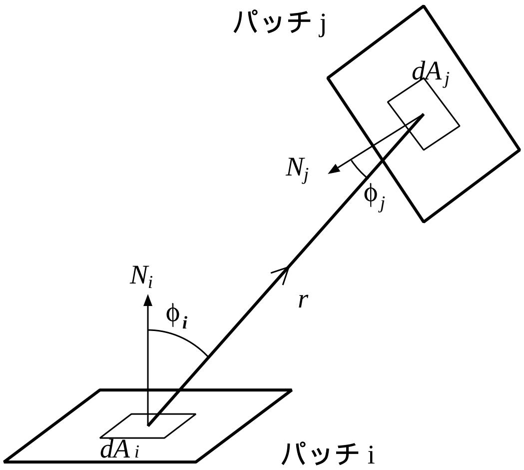


図 1　フォームファクタ

3.3　ヘミキューブ法によるフォームファクタの計算

この方法は，仮想的なヘミキューブ（半立方体）を用いることから，このように呼ばれている [5]．まず，図 2 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，この表面を適当な大きさのメッシュに分割する（図 2 の場合の分割数は 8 である）．そして，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する．そして隠面消去を行い，最も距離が小さい物体のみを残す．すると，F_{ij} は投影されたメッシュのフォームファクタ ΔF の和 $(\Sigma\Delta F)$ で求められる．

なお，本研究ではパッチをヘミキューブに投影するのではなく，ポリゴンを投影して隠面消去を行った後，各メッシュがポリゴン内部の，どのパッチに対応するかを計算する方法を用いる．これにより，パッチへの分割数を大きくした場合でも，投影と隠面消去を高速に行うことができる．

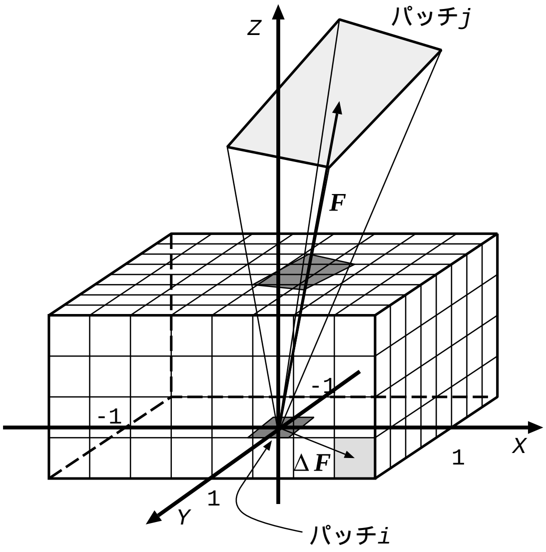


図 2　ヘミキューブ

3.4　漸進法

式 (2) は，各パッチの光の入射に注目しているが，逆に，各パッチの光の放射に注目し，他のパッチへの放射を繰り返すことで，全パッチのラジオシティの平衡状態に徐々に近づけていく方法である [6]．

入射に注目した式 (2) に対して，放射に注目すると，以下のような式ができる．

$$B_j = B_j + B_i(\rho_j F_{ji}) \quad (j = 1, 2, 3, \ldots, n) \tag{5}$$

この式を，ラジオシティの大きいパッチ i から順に適用することで徐々に解に近づけていくことができ，これを収束するまで繰り返す．従来の連立方程式を解く方法では，計算の最中は式 (3) にある全フォームファクタを使用する必要がある．一方，この方法では一度に使用するフォームファクタは式 (5) のように行列の縦 1 列分のみでよい．このためメモリを大幅に節約することが可能である．トランスピュータシステムではメモリ容量が比較的

小さいため，本研究では，この方法を用いる．

4　ヘミキューブの分割によるラジオシティ法の並列化

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．したがって，フォームファクタ計算の並列化が重要である．本研究では，ヘミキューブの 5 面のメッシュを，図 5 に示すようにラインごとに分割しておき，ポリゴンをヘミキューブに投影する処理の際に，各プロセッサに割り当てて Z バッファ処理を行わせる方法を用いる [7]．トランスピュータ 64 台の結合網は図 3 に示すようなトーラス結合を用いる．各プロセッサでの計算後，全対全通信を行い，全プロセッサが完全なフォームファクタを得る．

なお，パッチのデータは各プロセッサに分散して持たせている．漸進法による計算の際には，図 4 のように，まず各プロセッサ内で最大のラジオシティを持つパッチを決定し，次に，このデータを 1 箇所に回収して全パッチの中で最大のラジオシティを持つパッチを決定する．そして，このデータを全プロセッサにブロードキャストする．その後，フォームファクタの計算と通信を行い，各プロセッサが自分の保持するパッチへの放射の処理を行う．この方法の利点は，1 台ではメモリ不足で処理できないような複雑な画像でも，プロセッサの台数を増やすことにより処理が可能となることである．

4.1　静的負荷分散

ヘミキューブを分割したものを，各プロセッサに等分割する．ヘミキューブの各プロセッサへの割り当ては，物体の配置に偏りがある場

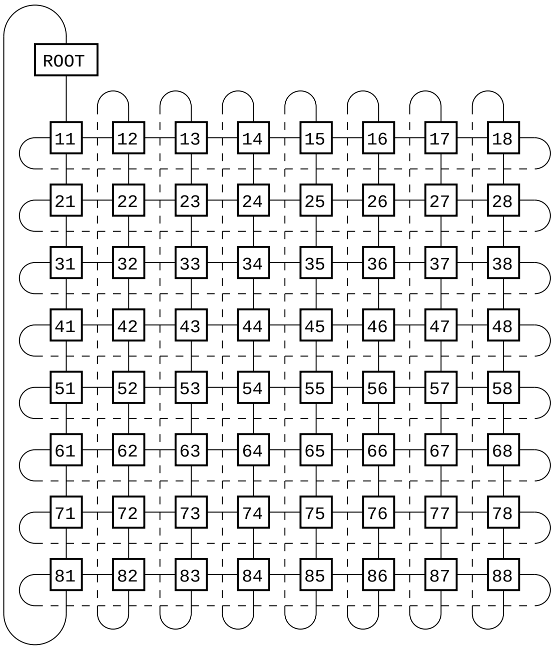


図 3　64 台トーラス結合

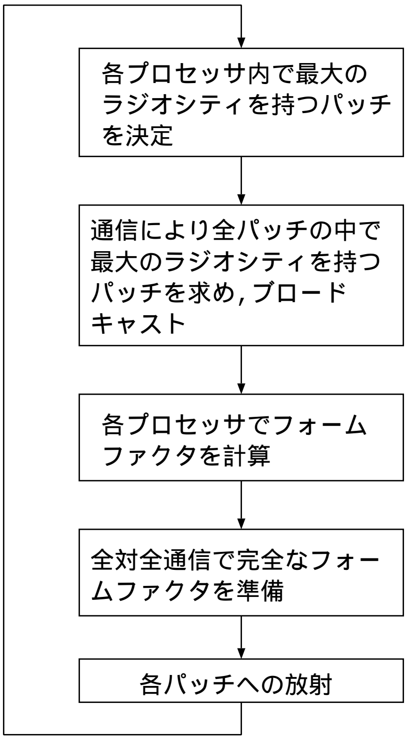


図 4　漸進法によるラジオシティ法の並列化

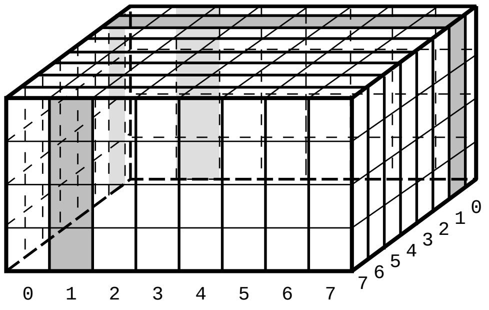


図 5　ヘミキューブの分割

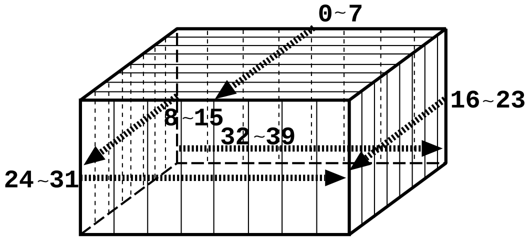
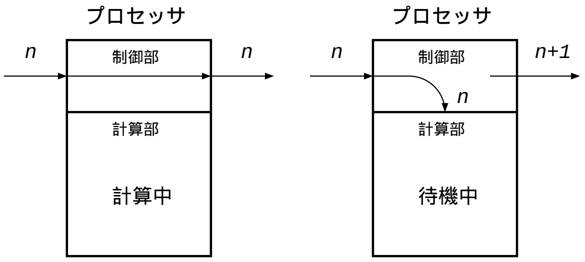


図 6　動的負荷分散での処理の割り当て順序



(a)計算中の場合　　(b)待機中の場合

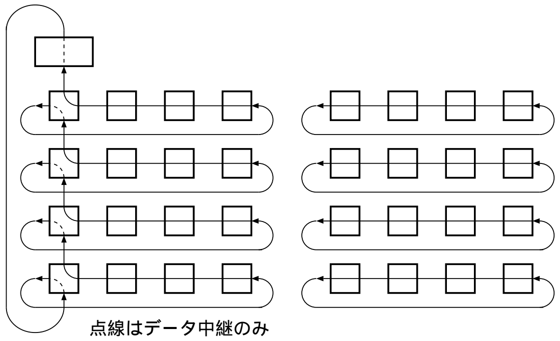
図 7　動的負荷分散の方法

合でも，なるべく各プロセッサに均等に負荷が分散できるようにする．具体的には，プロセッサを 0 から $T_n - 1$ までの T_n 台とすると，ヘミキューブのライン l は，$(l \bmod T_n)$ 番のプロセッサが計算を担当する．たとえば，分割数 8 のヘミキューブをプロセッサ 8 台で並列化する場合，図 5 の網掛部分全部を 1 台のプロセッサが担当することになる．

4.2　動的負荷分散

処理の割り当てに際して，ヘミキューブを (ヘミキューブの分割数)$\times 5$ の処理に分割し，図 6 に示すような順で処理番号を振っておく．この図のように分割数 8 の場合，上面が $0 \sim 7$，側面が $8 \sim 31$ となるようにする．これは，処理時間の大きいヘミキューブの上面の割り当てを最初に行い，その後側面を割り当てることで，処理をできるだけ均等に分割できるようにするためである．各プロセッサでは，図 7 のように，制御部，計算部の 2 つのスレッドを生成する．そして，各プロセッサの制御部の間でトークンを流す．トークンには，割り当てを行う処理番号と終了フラグを含んでいる．処理番号は 0 から開始し，もしプロセッサが待機中の場合はその処理番号を計算部に伝えると同時に，1 を加えたものを次のプロセッサへ送り出す．もし処理中の場合は処理番号をそのまま次のプロセッサに送り出す．このようにして，処理の空いているプロセッサに次々と処理を割り当てていく．終了フラグは，ヘミキューブの全ての部分の割り当てが終了したことを意味し，各プロセッサに全体全通信の開始を指示する．

動的負荷分散法においてプロセッサ数が多くなると，一つの処理が終わった後，次の処理が到着するまでに間隔が空くようになる．その対策として，グループ分割による動的負荷分散を用いる．グループ分割を用いる場合，トーラスの横 1 列のプロセッサを一つのグループとして，そのグループ内で動的負荷分散を行う．各プロセッサの制御部は図 8 に示すように処理番号を含むトークンの通信を行う．な



(a)グループ分割無し　　(b)4グループ分割

図 8　グループ分割による動的負荷分散

お，グループ分割による負荷分散の場合，各グループへの処理の割り当ては等分割に行う．つまり，グループ数をプロセッサ数とみなした静的負荷分散とする．グループ分割の場合，図 7 の処理番号に 1 を加える処理においてグループ数を加えることになる．

5　並列化の実験と考察

5.1　実験

静的負荷分散，動的負荷分散（グループ分割無し），動的負荷分散（グループ分割）の 3 通りの並列化方法で，ラジオシティ法による画像生成を実行する．トランスピュータシステムのプロセッサ台数は $16(4\times4)$, $32(4\times8)$, $64(8\times8)$ とし，それぞれ 1 台の場合の実行時間と比較する．ここでの実行時間とは，データを全プロセッサに送るところから始め，計算を行い，結果を ROOT に回収し終わるまでの時間とする．実行する内容は，表 1 に示すような 2 通りの画像において，漸進法の放射を 100 回行う処理を用いる．ヘミキューブの分割数は 128 とする．

表 1　ラジオシティ法で生成する画像内容

画像	ポリゴン数	パッチ数
1	192	894
2	372	1614

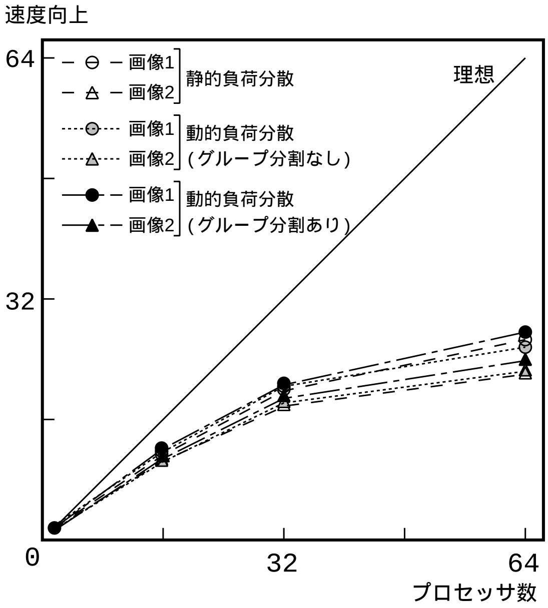


図 9　ラジオシティ法の並列化の速度比

5.2　考察

並列化の結果を図 9 に示す．図 9 より，どちらの画像においてもグループ分割による動的負荷分散が最も性能が良いことがわかる．また，グループ分割を行わない動的負荷分散でプロセッサが 64 台の場合，画像 1 の速度向上が静的負荷分散よりも悪くなっている．しかし，グループ分割を行うことで静的負荷分散法よりも良くなることがわかる．

本研究で用いた方法は，ヘミキューブ面を各プロセッサに分散させておき，各プロセッサは自分の担当する部分についてポリゴンの描画を行う．この際，ヘミキューブ面の場所に

表 2　負荷分散法の比較 (プロセッサ 64 台)

画像	負荷分散	各プロセッサの処理時間		平均トークン待ち時間 (ms)	静的に対する処理時間比
		平均 (ms)	最大 (ms)		
1	静的	183.1	199.4	—	1
	動的	195.3	204.7	1644.1	1.02
	グループ	180.4	191.5	88.4	0.98
2	静的	386.0	428.5	—	1
	動的	413.6	426.6	2428.6	1.00
	グループ	383.0	397.4	113.6	0.96

よって，投影されるポリゴン数に大きな偏りがあるため，各プロセッサごとの処理時間が不均衡になる．表 2 では，1 回の放射あたりの各プロセッサの処理時間の平均と最大を示している．ここで，フォームファクタの計算時間は描画処理時間が最大のプロセッサによって決定されるため，最大値が小さい負荷分散法ほど処理時間が小さくなっている．

また，プロセッサが多い場合には，グループ分割により効率が改善できることがわかる．表 2 の放射 1 回あたりの平均トークン待ち時間に示すように，動的負荷分散法では，一つの処理が終了して次のトークンが到着するまでの間，処理に空きが生じる．グループ分割を行わない場合に比べ，グループ分割を行った場合この差を小さくできるため，グループ分割が有効である．

6　おわりに

本論文では，トランスピュータシステム上でラジオシティ法による画像生成を並列化する方法とその効果を述べた．

結果として，プロセッサ 64 台の場合，1 台に対して約 20 ~ 25 倍の速度向上が得られた．静的負荷分散法と 2 通りの動的負荷分散法の比較では，プロセッサ台数が多い場合はグループ分割による動的負荷分散法が有効であることがわかった．

ただし，プロセッサ 64 台の場合，速度向上は理想値の 1/3 程度に留まっている．今後は，各部の処理時間を分析し，さらに並列化方法を改善する必要がある．

また，本実験では問題サイズを一定として，プロセッサ台数を変えて速度向上を評価した．並列処理の評価法として，メモリ制約や時間制約の方法も提案されており [8]，多様な観点からの評価を行って，最善の評価方法を見出す必要があろう．

参考文献

[1] C. M. Goral, K. E. Torrance, D. P. Greenberg, B. Battaile: “Modeling the Interaction of Light Between Diffuse Surfaces”, *Computer Graphics(Proc. of SIGGRAPH ’84)*, Vol. 18, No. 3, pp. 213–222(1984).

[2] 西田友是, 中前栄八朗: “陰および拡散照度を考慮した 3 次元物体の陰影表示”, 情報

処理グラフィックスとCAD研究会, 14–2, pp. 1–8(1984).

[3] 鷺島敬之, 西澤貞次, 浅原重夫: “並列図形処理”, コロナ社 (1991).

[4] 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”, グラフィックスとCADシンポジウム, Vol. 93, No. 6, pp. 95–104(1993).

[5] M. F. Cohen, D. P. Greenberg: “The Hemi-cube : A Radiosity Solution for Complex Environments”, *Computer Graphics(Proc. of SIGGRAPH ’85)*, Vol. 19, No. 3, pp. 31–40(1985).

[6] M. F. Cohen, S. E. Chen, J. R. Wallance, D. P. Greenberg: “A Progressive Refinement Approach to Fast Radiosity Image Generation.”, *Computer Graphics(Proc. of SIGGRAPH ’88)*, Vol. 22, No. 4, pp. 75–84(1988).

[7] 上嶋明, 山崎勝弘, 渡部透, 得丸英勝: “トランスピュータによるラジオシティ法の並列化”, 情報処理学会第48回全国大会論文集, No. 2, pp. 373–374(1994).

[8] J. P. Singh, J. L. Hennessy, A. Gupta: “Scaling Parallel Programs for Multiprocessors: Methodology and Examples”, *IEEE Computer*, Vol. 26, No. 7, pp. 42–50(1993).