卒業論文

テクスチャマッピングを用いた画面分割レイトレーシング法の並列化

氏　　名：柏原 正人
学籍番号：2210940087-8
指導教員：山崎 勝弘 教授
提 出 日：1998年2月20日

立命館大学 理工学部 情報学科

内容梗概

本研究では、並列計算機用のマイクロプロセッサであるトランスピュータを用いて、計算量の多い画像の並列処理による処理速度の向上を図った。その画像の生成法として、現在まで、本研究室において並列化が行われてきたレイトレーシング法に、よりリアルな画像の並列化を目的としてテクスチャマッピングの機能を追加した。

テクスチャマッピングは、模様を貼り付けることで、より現実感のある画像を表現できる手法であり、用いるテクスチャは、アルゴリズムによって生成するものと、写真などから取り込むものの2つに分類される。本研究では、拡張性の高さと計算量の多さからアルゴリズムによって生成する方法を用いた。

その並列手法に、レイトレーシング法の画素単位に独立して計算できる性質を活かした画面分割法を用いた。これより、テクスチャマッピングを用いたレイトレーシング法の評価を行った。

目 次

1　はじめに

マルチメディアの普及に伴い、コンピュータは文字情報だけでなく、図形、画像情報を扱うことが一般的になってきた。図形、画像をコンピュータで処理し、ディスプレイなどに映し出す技術をコンピュータグラフィックスというが、これまでの文字情報処理に比べてデータ量の多さ、複雑さにおいて処理の規模が膨大になるという特徴がある。[3]

レイトレーシング法は、光線追跡法ともいわれ、光の屈折、反射など光の複雑な現象を忠実にシミュレートした手法のひとつであり、CG の分野ではよく用いられている。視線が物体と交差する点を探索して、その色を画素の色とする方法であるが、画素の数だけこの処理を繰り返す必要があるので、高画質の画像を生成するには膨大な計算量を必要とする。[3][4] よって本研究では、画像処理における高速化技法について実現する。また、コンピュータグラフィックスを生成する方法として、さまざまな方法があるが、その中でも、レイトレーシング法についてとりあげ、よりリアルな画像を生成する技法として、テクスチャマッピングについてとりあげる。

また、テクスチャマッピングは、形状の表面に見られる模様を表現する方法である。本研究では、レイトレーシング法で生成した 3 次元物体上にテクスチャマッピングを適応させる。テクスチャマッピングにおけるテクスチャの生成法は、写真などの画像を直接利用する方法と、アルゴリズムによってテクスチャを生成する方法がある。前者の方法は、写真などのもつリアリティをそのまま生かせるという点で有用な方法であるが、視点や光源の移動に対しては対応が難しく、不自然な画像が生成されることが多い。後者の方法は、パラメータなどの変更によって、類似したテクスチャへの拡張が可能なものが多く、その有効範囲は広い。しかし、アルゴリズムによっては、膨大な計算量を必要とする。[4] 本研究では、後者の方法を利用する。

レイトレーシング法の高速化技法として、並列計算法、外接領域法、空間分割法などがある。並列計算法は、レイトレーシング法の、画素単位に独立して処理を行うことができる性質を利用して、複数の CPU に処理を分散させるハードウェアによる方法である。[6] 本研究では、英国の INMOS 社で開発された、並列計算機であるトランスピュータを用い、高速化を図った。トランスピュータは、 4 本のシリアルリンクを持ち、マルチスレッドにより並列処理を行う。CPU 間通信は 20MB/S で行う。

並列化において、あらかじめ画面を分割し、その部分を各プロセッサに振り分ける。計算の終了したプロセッサより値を返し、最も計算時間のかかったプロセッサでの計算時間が最終的にその画像生成における計算時間となる。その際、負荷均衡を考慮する必要がある。負荷均衡を考慮した方法として、インターリーブ分割がある。インターリーブ分割は、

例えば１６台のプロセッサを使用する場合、画面を４×４のブロックに分割し、ブロック内のピクセル１つ１つにプロセッサを割り当てるという方法である。[7] 本研究では、これまで行われていた、インターリーブ分割を用いたレイトレーシング法にテクスチャマッピングの機能を追加し、よりリアルな画像、そして計算量の多い画像を生成できるようにし、それをトランスピュータ複数台で処理した。また、テクスチャマッピングの部分にどれだけ負荷がかかっているかを計測するために、テクスチャマッピングを行っていない画像との処理時間の比較も行った。

本論文では、まず第２章でレイトレーシング法の特徴、第３章でテクスチャマッピングの特徴を述べ、また、マッピング手法、テクスチャの生成法について述べる。４章ではマルチトランスピュータシステムを用いたレイトレーシング法の並列化と画面分割について説明し、５章でその実験結果と考察について述べる。

2 レイトレーシング法

物体を観察するとき、物体表面に光源からの光線が当たって、反射、透過、屈折などの現象が起こる。光線は物体に当たるごとに物体の色がついてきて、目に入って像を結ぶ。よって、光源から出る光線を追いかけ、目に入ってくる光線を見つけ出すと、目に写る像が再現できる。しかし、光源から出た光線は、ほとんどが物体で吸収されるので、目に入ってくるものは少なく、１本ずつ追いかけたのでは効率が悪い。

そこで、光線を逆にたどることを考える。すなわち、視点からある方向に出た視線が、スクリーン上の点を通り、物体に当たり、反射、透過、屈折を繰り返す。そして、物体の色の情報を加えることにより、スクリーン上の点の色を決定する。このような画像生成の方法を、レイトレーシング法（光線追跡法）と呼ぶ。[3]
レイトレーシング法の基本的な手順は以下の通りである

1. 視点とスクリーン上を通る光線について、それと交差する物体を調べる。
2. 交差する物体があれば交点を求め、なければその画素を背景の色にする。
3. 交点のうちで最も視点に近い物体を選び、その点での輝度計算を行う。

また、以下にレイトレーシング法の特徴を示す。

- 比較的簡単なアルゴリズムで画像を描くことができる。
- 対象となる物体は基本的に関数で表現されなければならない。
- 光線を１本１本追跡するので演算に時間がかかる。

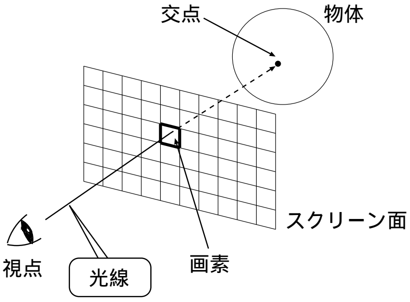


図 1: レイトレーシング法

実際の物体を立体的に見せるためには、陰影や付影が重要な役割を果たす。この陰影や付影のような現象を表現するために、以下に示すような要素を考慮して計算する。

- 周囲の環境光による輝度

現実世界の環境では光源から物体に光が投射されている場合、直接光のあたっていない部分でもぼんやりと観察することができる。これは、周囲の物体からの散光や間接光により、一定の光の明るさがあると考えられる。

$$\text{一定の光の明るさ} = \mathrm{L}_a$$

- 拡散反射光による輝度

光源からの入射光による反射光のうち、その一部はすべての方向に一様に散乱されているものを拡散反射光とする。これによる光の強さは以下に示すように、光源の入射角の余弦に比例する。

$$\mathrm{L}_d = K_d \cdot I_p \cdot \cos\theta$$

で表される。ここでI_pは入射光の強度、K_dは物体特有の拡散反射係数である。

- 鏡面反射光による輝度

光源から物体に照射された光は表面が金属とか鏡のような材質でできている場合には強く反射する。すなわち、視点が反射角の位置にある時に最も強度が大きくなり、この位置から離れる角度 ϕ にしたがって光の強さが低くなる。この場合の光の強さは、視点の位置により異なり、以下に示すような式になる。

$$\mathrm{L}_s = I_p \cdot w(\theta) \cdot (cos\phi)^n$$

ここで、wは物体特有の鏡面反射率であり、入射角θ の関数で与えられる。n はハイライトの度合いを決める値で 1 から 10 の値をとることが多い。

以上の輝度の要素を加えた次式によって面上の一点の光の強さを計算することができる。

$$L = \mathrm{L}_a + \mathrm{L}_d + \mathrm{L}_s$$

3 テクスチャマッピング

3.1 テクスチャマッピングの特徴

マッピング (mapping) とは、「地図を作る」という意味のほかに、数学の「写像、座標変換」の意味がある。コンピュータグラフィックスの分野では後者の意味で、2次元の平面上にある絵柄や模様などを、3次元曲面や平面の上に貼り付けることをマッピングという。

ここで、貼り付ける模様のことをテクスチャといい、2次元テクスチャを3次元曲面などに写像することをテクスチャマッピング (texture mapping) という。この作業は2次元平面であるテクスチャの空間上の1点と、3次元物体の空間上の1点とを対応づけることによって行うことができる。

その対応づけの方法は以下のようなものがある。

1. 投影による方法
2. 座標変換を用いる方法
3. マッピングパラメータを曲面データに付加する方法

本研究では、2番の座標変換を用いる方法により、テクスチャマッピングを行った。その方法については、3.2で述べる。

またテクスチャマッピングは、写真などの画像を直接利用する方法と、コンピュータによってテクスチャを生成する方法の2つにわけられる。

画像を直接利用する方法は、テクスチャを貼ることにより、簡単にリアルな画像を生成することができる。しかし、ある条件下で撮影された写真などを直接用いるため、CGによる視点や光源の移動に対しては対応が難しく、不自然な画像が生成されることが多い。これは、視点や光源が、その写真が撮影されたときと異なった状況の3次元CGを作ろうとした場合に生じる問題で、単一の条件下での画像のみでは、これを回避することはできない。

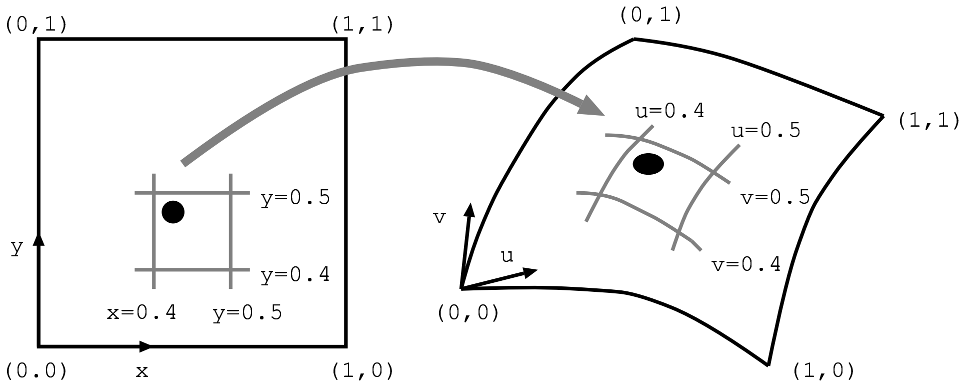
コンピュータによってテクスチャを生成する方法は、テクスチャ生成の手順を構成しておいて、コンピュータにテクスチャを自動生成させるものである。コンピュータの中にテクスチャの色や形状のデータをもっているので、任意の視点や光源位置でのCGを作ることができる。また、類似したテクスチャへの拡張が可能なものが多い。テクスチャの生成の過程で、乱数やフラクタルなどを導入することにより、生物特有の不自然さや、自然物

らしさの表現も可能である。[1][3][4][5]

上に述べた通り拡張可能であることや、計算量が多く、並列化に適していることから、本研究では、コンピュータによってテクスチャを生成する方法を用いた。

3.2　マッピング手法

2次元平面であるテクスチャの空間上の1点と、3次元物体の空間上の1点とを対応づけるために、3次元物体の表面の点の座標を2つのパラメータに変換する必要がある。例えば、表面の点を(u,v)の2つの座標に変換した場合、図2のようにテクスチャの(x,y)の2つの座標に対応づけることができる。[5]



テクスチャ　　　　3次元曲面

図 2: マッピングによる座標の対応

2つのパラメータに変換するために、平面は直交座標系、球面は緯度-経度座標系に変換すればよい。以下に、変換方法を述べる。

直交座標変換

図 3 で、平面 Π_3(a_1と a_2を通る平面) 上の点は、ある 2 個の実数で表すことができる。しかし、これは、 a_3には依存しない。ここで、a_3をとり除いたものを、直線座標系という。

このとき、a_1と a_2が直交し $|a_1| = |a_2| = 1$ を満たすとき直交座標系という。

要するに 3 次元空間上の平面は、 2 つのパラメータで表すことができる。例えば、画像を生成するとき、y 軸に垂直に平面を作りたい場合、y に関係ない x の座標を u,z の座標を v というパラメータに置き換えればよい。このパラメータとテクスチャの座標を対応づければ、平面へのテクスチャマッピングを行うことができる。

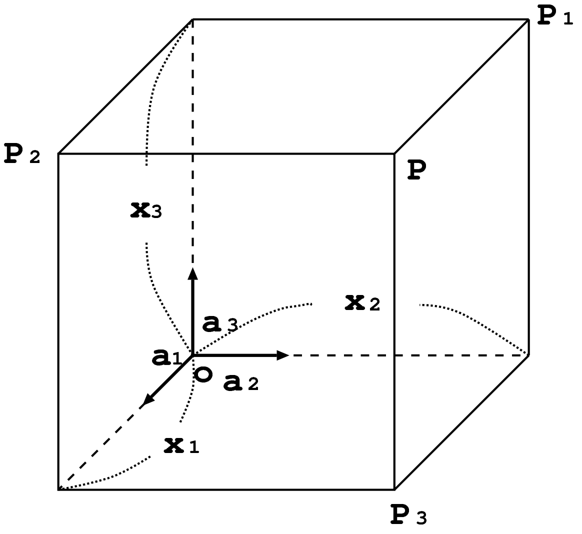


図 3: 直交座標変換

緯度-経度座標変換

緯度-経度座標変換は地球儀を作るという考え方である。要するに、何も貼られていない球体に、２次元平面である世界地図を自然な状態で貼るために、緯度-経度をパラメータとして用いるということである。

２次元平面の緯度-経度の値を球面に対応させるため、まず図 4 のように、球の中心座標と視線との交点の差を、x,y,z それぞれにおいて求める必要がある。その後、差である WX,WY,WZ より下の式を用いて緯度-経度座標に変換する。ここでは (u,v) を緯度-経度としている。

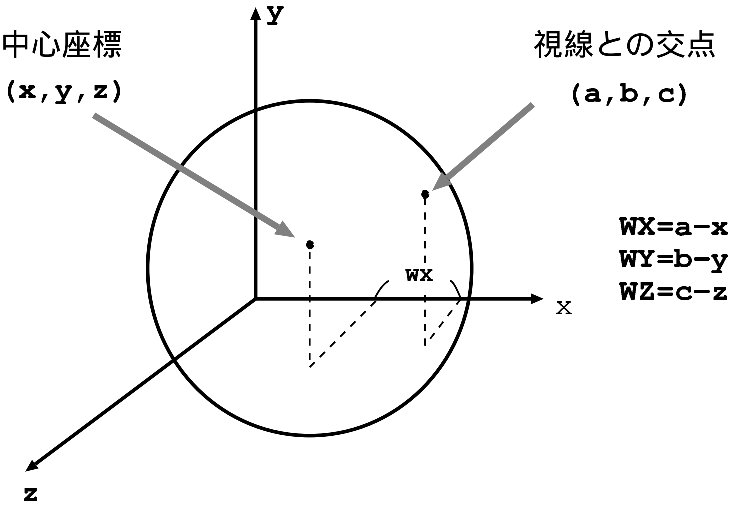


図 4: 緯度-経度座標変換

$$\mathrm{WXZ=WX*WX+WZ*WZ} \tag{1}$$

$$WX > 0 : u = arctan(WZ/WX) + \text{円周率}\times\ 1.5 \tag{2}$$

$$WX <= 0 : u = arctan(WZ/WX) + \text{円周率}\times\ 0.5 \tag{3}$$

$$WXZ < 0 : v = 0 \tag{4}$$

$$WXZ >= 0 : v = arctan(WY/\sqrt{WXZ}) + \text{円周率}\times\ 0.5 \tag{5}$$

3.3 テクスチャの生成法

コンピュータによりテクスチャを生成するには、そのアルゴリズムを考える必要がある。このアルゴリズムを変更することで多種多様の模様を生成することができる。

例えば、物体表面上の点が (u,v) で表されている時、図 5 のような市松模様を作る場合、式 (6) を用いればよい。ここで、d は市松模様のサイズを決定する定数であり、演算 a mod b は、a を b で割ったときの余りを与えるものである。要するに u,v ともに d/2 より小さいか、ともに大きい場合、色 1 で模様を生成する。また、u が d/2 より小さく、v が d/2 より大きい時とその逆の場合、色 2 で模様を生成する。この方法により 2 種類の色で市松模様を作ることができる。このようにして色が定まったならば、あとはレイトレーシングの計算にのせればよい。

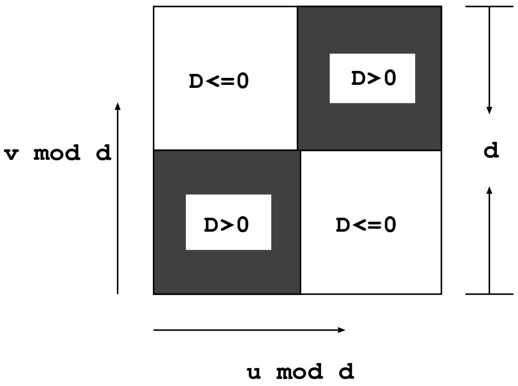


図 5: 市松模様の作り方

$$D=((u \bmod d)-0.5d)((v \bmod d)-0.5d) \quad (6)$$

$D > 0$: 色 1

$D <= 0$: 色 2

また、スキャナなどで入力して得られたデジタル画像は、

デジタル画像を `P(i,j), 0<=i, j<=M-1`

テクスチャの大きさを定める定数を `d`

とすると、点 `(u,v)` には画像のピクセル

`P((u mod d)M/d, (v mod d)M/d)`

の色を与えればよい。[1]

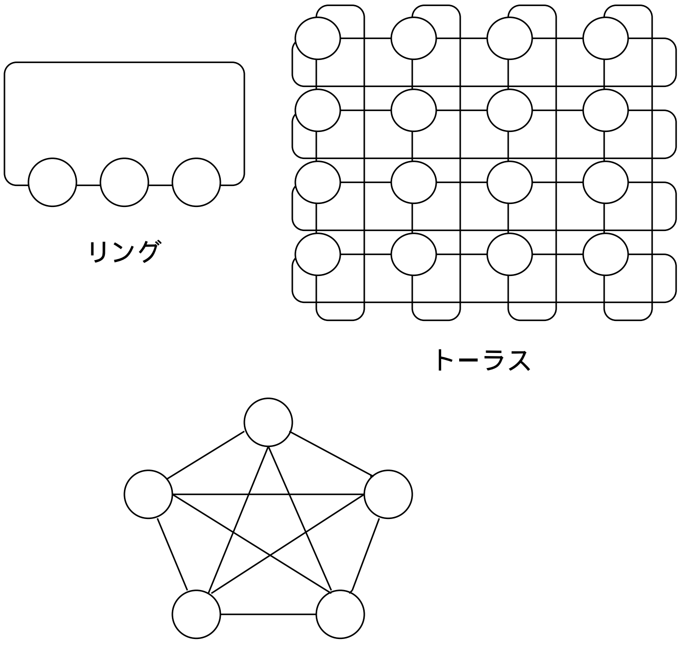
4　レイトレーシング法の並列化

4.1　マルチトランスピュータシステム

トランスピュータは、英国のINMOS社によって開発された並列計算機用マイクロプロセッサである。本研究室に設置されて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バイト備え付けられている。外部との通信はINMOSシリアルリンク(10Mbps × 4)で行われる。現在、64台のトーラス結合（図6参照）で接続されている。また、外部との入出力を行うためにルートプロセッサが1台ある。

使用する言語はParallel Cを用いる。これは一般のC言語にプロセッサ間通信に必要な関数などが拡張されている。Parallel Cによる並列計算では各トランスピュータに複数のタスクが存在し、タスク間は通信によりデータを交換することができる。その際、タスク間の通信は各チャンネルが1対1の関係で、かつ送受信の要求が一致した場合、転送が開始される。要求が片方のみの場合、もう一方の要求が出るまでウェイト状態となる。

トランスピュータの結合網には図6に示すようなものがある。[6]


リング

トーラス

完全網

図 6: 相互結合網

4.2 画面分割

レイトレーシング法では、画面上の各ピクセルを独立に計算することができる。それを利用してスクリーンを分割し、複数プロセッサで並列処理をすることで高速化を図る。そこで、図8のように画面を実行可能なプロセッサの数に分割し、並列計算を行う。

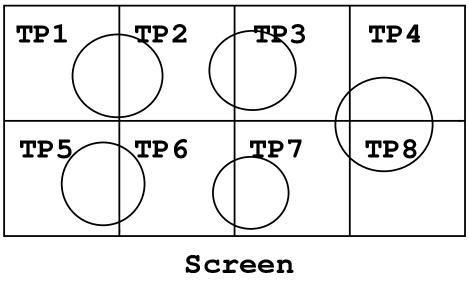


図 7: 画面分割 （8台の例）

4.3 インターリーブ分割

あるピクセルの周りのピクセルも、ほぼ同じ負荷であるという考えに基づいた負荷均衡を考慮した方法である。図8はトランスピュータ１６台を用いて並列化を行う場合、分割された画面のピクセル１つ１つに、トランスピュータ１台をそれぞれ割り当てている。これにより、ほぼ同じであると考えられる負荷をトランスピュータに割り当て、処理させている。

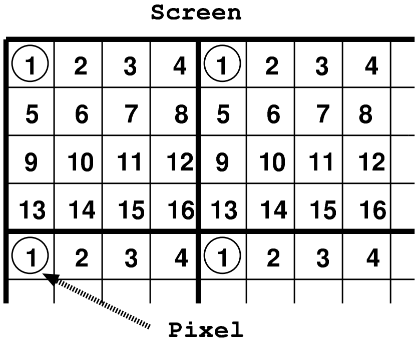
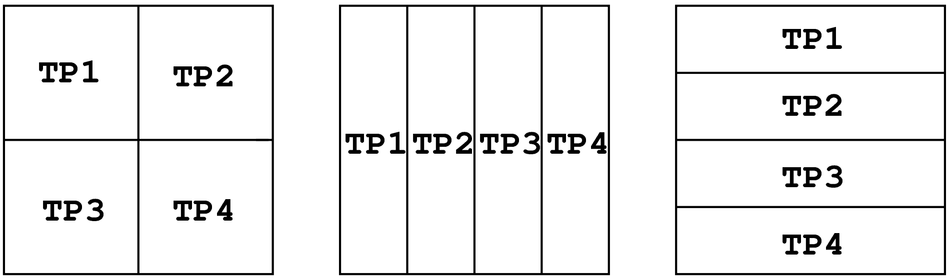


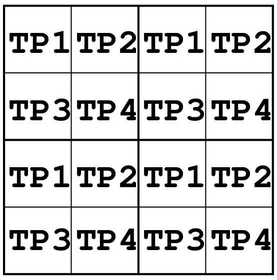
図 8: インターリーブ分割

4.4 マルチトランスピュータによる並列化

本研究では、画面等分割 (メッシュ、縦、横)、インターリーブ分割についてリング結合により並列化を行った。



メッシュ等分割　縦等分割　横等分割



インターリーブ分割

図 9: 画面分割方法

画面等分割は、プロセッサが n 台のとき、画面を n 個に分割し、各プロセッサにその範囲の計算をさせる方法である。各プロセッサには、あらかじめ分割された範囲の左上座標と右下座標を与え、処理がすみ次第、結果を Root に送り返す。また、本研究においては、画面の大きさを 640 × 400 にしているため、横等分割の際、用いるプロセッサの台数で画面のサイズが割り切れない時がある。その場合、余ったピクセルを 1 ピクセルずつ適当なプロセッサに振り分ける。

インターリーブ分割は、分割された画面内の 1 ピクセルごとにプロセッサを振り分けている。画面がきれいに分割される必要があるので、用いるプロセッサ数は限定される。

各プロセッサは、メモリとして 1MB をもっている。各プロセッサとも同じ数のピクセルの処理を行うため、1 プロセッサ当たり (640 × 400)/64=4000 ピクセルを受けもつことになる。ピクセルは RGB(各 1 バイト) の値とトランスピュータが 32 ビットであることから、各プロセッサのデータ領域は 4000 × 4=16KB 必要となる。

5　実験と評価

5.1　実験方法

実験はトランスピュータを用いて、画面等分割（メッシュ、縦、横）、インターリーブ分割の実行時間を求める。用いたトランスピュータの台数は、1,4,8,16,32,64 台でそれぞれにおいて実行時間を求める。またその評価方法として、相対速度を用いる。相対速度は、下の式の通りでトランスピュータ 1 台の時と比較して、どの程度処理時間が減少しているかを計るものである。

$$\text{相対速度} = \frac{1\,\text{台トランスピュータによる実行時間}}{n\,\text{台のトランスピュータによる実行時間}}$$

用いる画像は、画面等分割の精度を見るために、画面に均等な分布のものと、画面に均等でない分布のものを用意する。また、テクスチャの負荷を確かめるために、画面に均等な分布のものにおいては、テクスチャマッピングを行っていない画像も用意する。

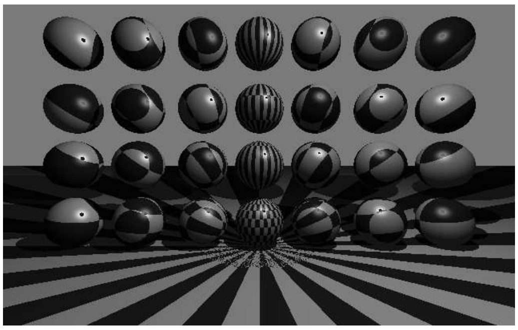


図 10: test1（画面に均等な分布）

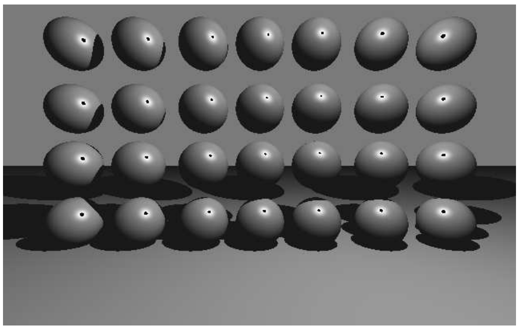


図 11: test2（テクスチャを貼っていない）

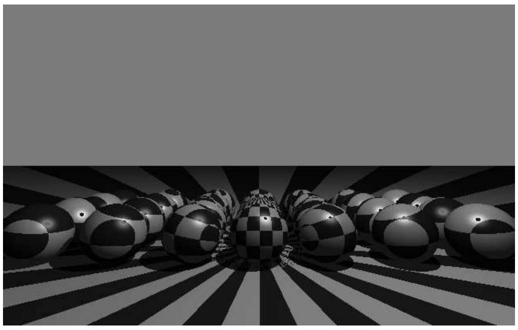


図 12: test3（画面に均等でない分布）

5.2　実験結果と考察

表 1: test1 の実行時間（秒）と相対速度

	1	4	8	16	32	64
メッシュ等分割	461.82	164.03	90.131	72.476	53.290	39.437
相対速度	1.0000	2.8154	5.1238	6.3720	8.6661	11.710
縦等分割	461.82	150.81	85.907	64.145	47.098	37.901
相対速度	1.0000	3.0622	5.3758	7.1996	9.8055	12.184
横等分割	461.82	148.75	83.541	61.079	49.201	40.614
相対速度	1.0000	3.1046	5.5280	7.5610	9.3863	11.370
インターリーブ分割	461.82	140.12	75.531	57.214	40.652	30.245
相対速度	1.0000	3.2958	6.1143	8.0718	11.360	15.269

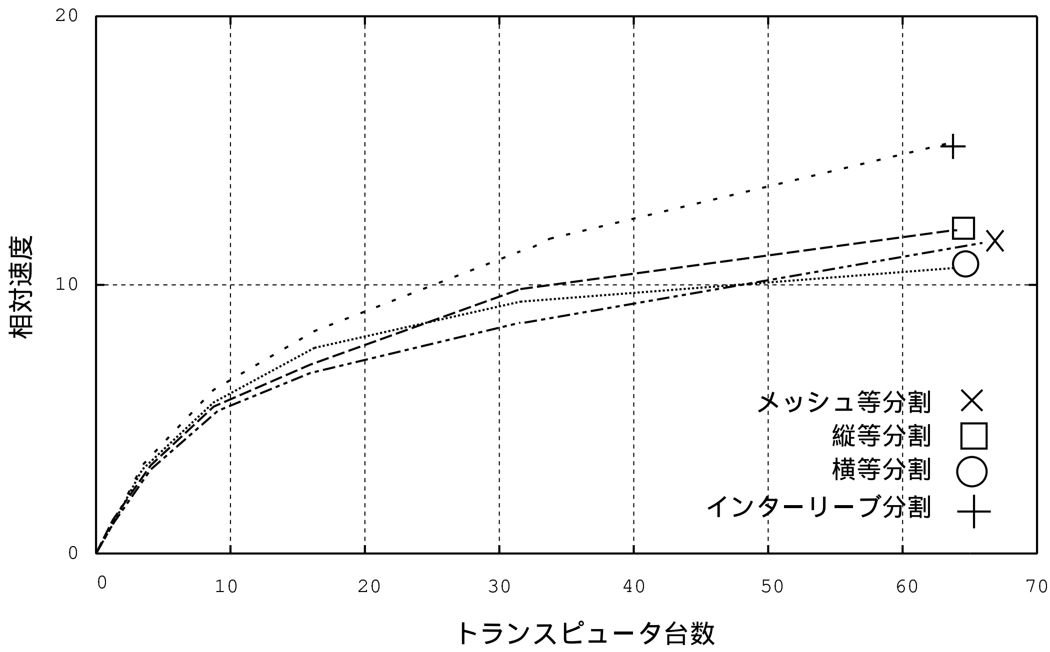


図 13: test1 の各分割による相対速度

表 1 は、トランスピュータ 1 台から 64 台の各分割による実行時間と相対速度である。図 13 は、相対速度とトランスピュータの関係をグラフにしたものである。

test1 の画像は、1 台で処理するのに 461.82 秒かかるのがわかる。

表2: test2の実行時間（秒）と相対速度

	1	4	8	16	32	64
メッシュ等分割	430.42	155.76	86.781	70.011	52.390	39.274
相対速度	1.0000	2.7633	4.9598	6.1478	8.2159	10.969
縦等分割	430.42	142.98	82.769	63.089	46.453	37.669
相対速度	1.0000	3.0103	5.1985	6.8224	9.2657	11.426
横等分割	430.42	138.59	79.004	59.234	49.964	40.326
相対速度	1.0000	3.1057	5.4480	7.2653	8.6146	10.673
インターリーブ分割	430.42	133.90	72.554	56.093	39.851	30.062
相対速度	1.0000	3.2144	5.9324	7.6733	10.800	14.317

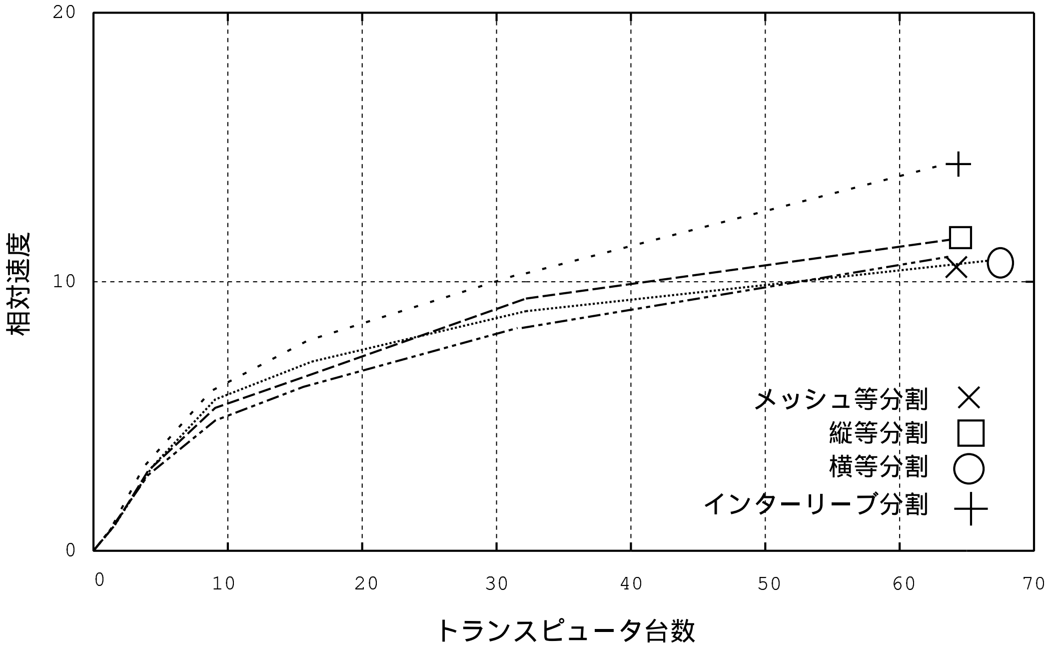


図14: test2の各分割による相対速度

表2よりtest2の画像は、1台で処理するのに430.42秒かかるのがわかる。これと表1より、テクスチャマッピングの処理におよそ30秒かかっていることがわかる。

表 3: test3 の実行時間（秒）と相対速度

	1	4	8	16	32	64
メッシュ等分割	405.03	148.26	84.264	68.921	50.983	38.765
相対速度	1.0000	2.7318	4.8066	5.8767	7.9444	10.448
縦等分割	405.03	142.76	77.869	62.392	45.861	37.023
相対速度	1.0000	2.8371	5.2014	6.4916	8.8316	10.939
横等分割	405.03	147.48	83.289	65.530	48.643	40.263
相対速度	1.0000	2.7463	4.8629	6.1808	8.3265	10.059
インターリーブ分割	405.03	135.01	71.417	55.961	39.042	29.692
相対速度	1.0000	3.0000	5.6713	7.2377	10.374	13.641

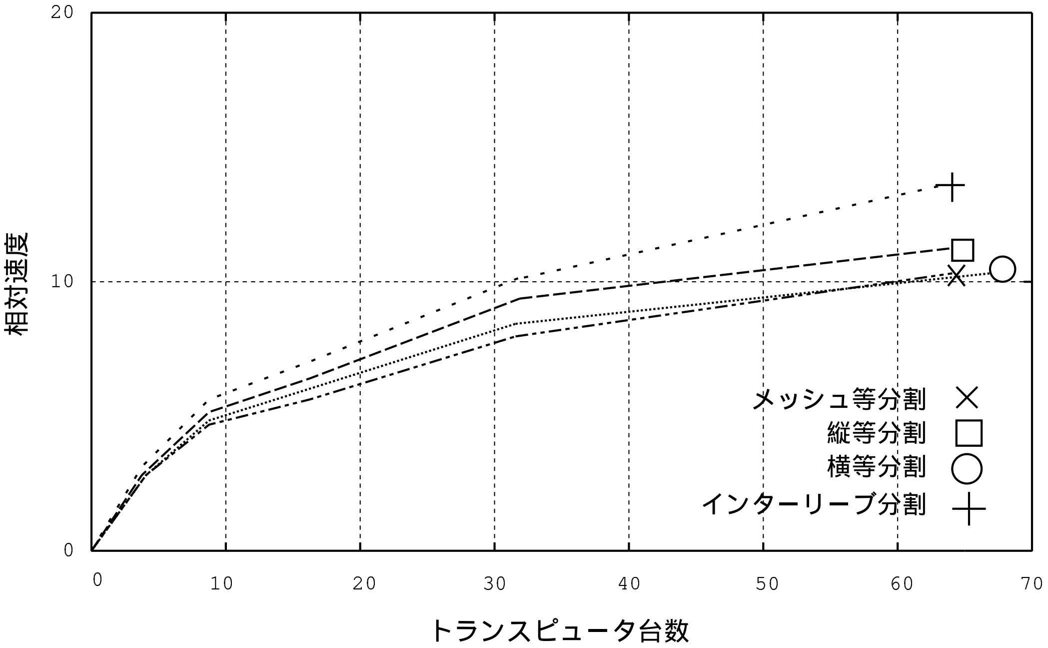


図 15: test3 の各分割による相対速度

表 1,2 に比べ、縦等分割より横等分割の方が精度が下がっていることがわかる。これは、負荷が横に広がっているからと考えられる。

すべての画像をとおして言えることは、従来研究されていた通りに、インターリーブ分割が最も性能が良いことがわかる。

テクスチャを貼っている test1 の画像とテクスチャを貼っていない test2 の画像の処理結果を比較すると、トランスピュータ 1 台での処理時間がそれぞれ、461.82 秒と 430.42 秒であることから、テクスチャマッピングの処理におよそ 30 秒かかっていることがわかる。また、トランスピュータ 64 台での処理時間をそれぞれ、比較すると各分割において、テクスチャを貼っていない方が、0.2 から 0.3 秒ほど速くなっていることがわかる。よってトランスピュータ 64 台では、テクスチャマッピングの処理に 0.2 から 0.3 秒かかっている。これはテクスチャマッピングの処理の部分においての相対速度で、100 から 200 倍の性能向上が得られたという結果である。プロセッサの台数倍の性能向上が得られるという、理想値からいえば、はるかに性能向上が得られている。しかし、テクスチャマッピングを含む全体の処理から言えば、一番性能の良いインターリーブ分割でも、15 倍程度の相対速度しか得られていない。これは、処理の際の通信に時間が、かかりすぎているためと考えられる。これは、トランスピュータの結合網を、リング結合以外のもので行えば、解決されるものと考えられる。

また画面に均等な分布である test1 の画像と均等な分布でない test3 の画像の処理結果を比較すると test3 は、横等分割より縦等分割の方が、性能が良いことがわかる。これは、test3 の画像が、横に負荷が分散しているために、プロセッサに負荷を均等に振り分けるためには、縦等分割の方が効率が良いことがわかる。

またすべての画像の処理結果で横等分割の相対速度が、トランスピュータ 32 台から下がっているのは、画面の横のサイズが 400 であるため、32,64 で割り切れない部分を、適当に分散させているので、通信速度が普通に比べ多くかかっていることが原因であると考えられる。これは分散の方法をもう少し考慮することで、相対速度を上げることができると考えられる。

6　おわりに

本研究では、テクスチャマッピングを用いたレイトレーシング法を、画面分割によりトランスピュータを用いて並列化を行った。レイトレーシングのアルゴリズムは簡単で、しかも各ピクセルの計算が独立しているために並列化に向いている。テクスチャマッピングは、そのアルゴリズムにより、いろいろな模様を表現できる。今回は単純な模様で実験を行ったが、計算量の多い別の模様で実験を行う必要があると思われる。

並列化においてテクスチャマッピングの部分は、並列効果がみられているが、全体的には、理想値よりも下である。これは、トランスピュータ間のデータの通信時間によるものと考えられる。今回用いたトランスピュータの結合網はリング結合で、データの通り道は一方のみであることから、全体的な処理時間も多くなっていると考えられる。

並列手法に画面等分割を用いたが、分割方法はインターリーブ分割が、最も性能が良いことがわかる。他の分割方式は画像により、有利な方法があると思われ、その順位づけはできない。

謝辞

本研究を行うにあたり、その機会を与えて下さり、数々のご指導、または助言をいただきました山崎勝弘教授に、心より深く感謝の意を表します。

また、本研究を進めていくうえで、いきづまった時など、いろいろな面で貴重なご意見をいただきました本研究室の皆様に心より感謝致します。

参考文献

[1] 千葉則茂 , 村岡一信, “C による CG レイトレーシング”, サイエンス社 (1991),pp.9-44.

[2] 杉原厚吉, “グラフィックスの数理”, 共立出版社 (1995),pp.1-80,169.

[3] 小堀研一 , 春日久美子, “基礎から学ぶ図形処理”, 工業調査会 (1996),pp.111-116.

[4] 安居院猛 , 中嶋正之, “コンピュータグラフィクス”, 昭晃堂 (1992),pp.89-101.

[5] Andrew S.Glassner, “最新三次元コンピュータグラフィックス”, アスキー (1991),pp.137-152.

[6] 山本正樹 , 中井泰明 , 村上安範, “トランスピュータ入門”, 日刊工業新聞社 (1990)

[7] 吉谷崇史, “画面分割によるレイトレーシング法の並列化”, 立命館大学理工学部情報工学科卒業論文 (1995).

[8] 吉谷崇史, “空間分割によるレイトレーシング法”, 立命館大学大学院理工学研究科情報システム学専攻修士論文 (1997).