卒業論文

空間分割レイトレーシング法を用いた透過物体の並列化

氏名：隅垣 孝徳
学籍番号：2210940183-1
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，並列計算機トランスピュータ上で，画像生成法として有名なレイトレーシング法の並列化を空間分割を用いて行った．今までに本研究室で行われたレイトレーシング法の並列化の研究を引継ぎ，さらに広範囲の性質を持つ物体を対象にした並列化を試みた．新しく対象にする物体として，透過物体を選択した．空間分割は，物体の配置された状況により，ボクセルの位置や大きさを変える空間分割である．

今までの並列レイトレーシングシステムに，透過物体を処理できる機能を実装することで，透過物体を表現できるようにした上で，以前の研究でも評価のために使用された 6 つの画像を対象に実験を行った．プロセッサの数を増やし，同時に処理することで得られる速度向上について実験，考察した．また，1 度に出す光線の数を変化させた時に，どういった速度変化が表れるかを実験，考察し，適切な本数を見いだした．

目 次

1 はじめに

コンピュータグラフィクス（Computer Graphics:CG）は，絵がコンピュータと人との対話に積極的に活用されるようになってから飛躍的に進歩してきた．CAD やゲームだけでなく，最近では TV や CM でも CG が取り込まれることは当然となり，映画の世界でも本格的に導入されて CG 映画が主流となりつつあるほどである．中でも映画やゲーム，CAD で用いられている 3 次元 CG（3D-CG）の発展は大変目まぐるしく，非現実的な状況でも非常にリアルに表現できるようになり，これからも期待が大きい情報技術であるといえる．

画像生成アルゴリズムの中でも特によく知られているのがレイトレーシング法である．レイトレーシング法は，物体表面で反射，屈折を繰り返す光を視点に届く光に限定し，視点から逆に追跡し，物体の輝度を決定する．このため比較的分かりやすいアルゴリズムであり，実世界の現象をかなり忠実にシミュレートしたものなので，非常にリアルな画像を表現できる．反面、光線は視点からスクリーン上の各画素を通るように出すので，画素数分の光線を出さなければならず，さらに物体交点で光線が反射，屈折するときにも光線の数が増え，それら全ての光線について交点計算を行うために計算量は膨大となる [1]．

その高速化技法として，ハードウェア的な複数のプロセッサでの並列計算による高速化や，ソフトウェア的なバウンディングボリューム法，画面分割や空間分割法の適用による高速化が上げられる [1]．特にレイトレーシング法は，画素ごとに出すそれぞれの光線が完全に独立して計算を行うことができるので，複数のプロセッサによる並列計算は大変効率が良い．本研究では，英国 INMOS 社で開発された並列計算機であるトランスピュータ上で，空間分割法によって交点計算量を削減し，プロセッサ間の負荷均衡を計ることにより高速化を実現している．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 で行われる [8]．

本研究では，これまでに作成された空間分割を用いた並列レイトレーシング法 [6] に，透過物体の処理系を追加することで，より広範囲の画像に対しての並列性を評価できるようにした．透過物体とは，例えばレンズのような物体で，光を通すことで後ろの物体の存在が確認できるものである．これには窓ガラスのように光の進行方向を変えないものと，虫眼鏡のように光の進行方向が変えられる（屈折する）ものがある．透過物体自身の陰影付け，透過物体が落とす影の実現と，屈折による光線の進行方向の計算など両性質の透過物体が表現できるように機能を付け加えた．さらに反射と屈折で枝分かれしていく光線を管理するために，通信パケットの変更も行った．今回の研究では，物体の配置状況によってボクセルの大きさや配置を決定する空間分割 [6] を用いて，屈折を含む透過物体の並列性と，同時に出す光線の本数による実行時間の変化について考察した．実験方法は，6 つの画像についてトランスピュータ 1，4，16，64 台で実行時間を計測した．また，それぞれの画像で最大の並列性が得られたプロセッサ数において，同時に出す光線の数が 1，8，16，

32，64，96，128 本の場合の実行時間を計測することで適切な光線本数を見いだし，拡散，鏡面物体時の適切な光線本数との比較考察を行った．

本論文では，2 章でレイトレーシング法の特徴，画像生成法，そして適応空間分割型レイトレーシング法について述べ，3 章で適応空間分割型レイトレーシング法の並列化について，トランスピュータシステムとそれによる並列計算手法を述べる．4 章では適応空間分割を用いて透過物体を並列化した場合の並列効果と，拡散，鏡面物体時のものとを比較し，考察を行う．5 章で本研究の成果と今後の課題を述べる．

2　レイトレーシング法

2.1　レイトレーシング法の特徴

人の目が物体を認識できるのは，物体から色のついた光が照射されるからである．しかし目に見える物体の多くは，その物体自身が発光しているのではなく，光源によって照明されることで自身の色を放射することができる．光源から出た光は，様々な物体に当たり反射や透過を繰り返し，そのうちのいくらかの光線により像が再現される．その間に，光はその物体の色を帯び，人の目に色を認識させる．よって，光源から照射される光線を1本ずつ追跡し，目に入ってくる光線を見つけ出すと，その光の色により像が再現できる．しかし物体に光が当たった時，その光はあらゆる方向へ照射されるのが常であり，また永久に反射、透過を繰り返すのではなく物体そのもので吸収されることがあるので，光源から照射された光線で目に入るものはごくわずかである．これを，1本ずつ追跡したのでは大変効率が悪い．そこでこれを逆に，目に入ってくる光線を視点から追跡することを考える（図1）．視点から，スクリーンの各点を通る光線を追跡し，当たった物体の色を付けながら反射，透過を繰り返すことでその点の色が決定される．スクリーン上の全ての点においてこの要領で処理を行うことで，目に写る物を再現できる．このような処理で画像を作り出す方法を「レイトレーシング法」と呼ぶ．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表現できる．
- 反射，透過など様々な性質の物体を表示できる．
- 光線を1本ずつ追跡するので演算時間がかかるが，各光線の処理が独立なので並列処理に向いている．

図 1: レイトレーシング法

2.2 透過物体の表示

ガラス球や透明のグラスなどの透明な物体では，光の一部は反射するが他の一部は透過する．この透過する光により，人の目は後ろの物体を認識することができる．この時，後ろ側に見える物体が歪んで見えることがある．これは透過物体内部と外部の屈折率の変化により，光の進行方向が変えられるからである．この屈折について説明する[1][4]．

光が進行してくる物体をa，入射して屈折させる物体をb，それぞれの入射角（法線からの角度）をθ_1 θ_2，それぞれの屈折率をn_1，n_2とするとき，

$$n_1 \sin\theta_1 = n_2 \sin\theta_2 \tag{1}$$

の関係になる．これを「Snell の法則」と呼ぶ．

図 2: 屈折光

異なる屈折率の界面に光が入射すると必ず一部は反射して，残りは屈折する．この時，入射角θ_1が大きければ大きいほど反射光量は多くなる（Fresnel の公式）．さらに屈折率の大きな物から小さなものへ光が向かう時，入射角が一定値以上のときは光が全て反射されるという現象が起こる．これを全反射と呼び，式（1）を変形したもの

$$\sin\theta_2 = \frac{n_1}{n_2}\sin\theta_1 \tag{2}$$

から$n_2 < n_1$により，θ_1の値によっては$\sin\theta_2$が1より大きくなってしまう．この時，光は屈折せずに全反射をする．式（2）の$\sin\theta_2$が1になるときの角θ_1を臨界角と呼ぶ．

ここで，屈折する光線の進行方向の求め方を説明する．

光は屈折率n_1の物体aから，屈折率n_2の物体bに進むものとし，界面にある点Oを始点にしたベクトルで考える．入射光線ベクトルをEとし，屈折して進行方向が変えられたベクトルを$E^{'}$とする．また，法線ベクトルをNと，入射光線が屈折されずに進んだ時のベクトルをRとし，Eの長さをn_1，$E^{'}$の長さをn_2とする．Nは単位ベクトルとする．

まず，ベクトルRの終点からベクトル$E^{'}$の終点に向かうベクトルAを考えると，

$$E^{'} = R + A \tag{3}$$

になる．光は密度の高い物質中ほど速く進み，2つの物質中の速さの比は比屈折率となるため，ベクトルAは法線ベクトルNと平行になる．

各ベクトルの方向余弦が，

$$R = (R_x, R_y, R_z)$$

$$E^{'} = (E^{'}_x, E^{'}_y, E^{'}_z)$$

$$N = (n_x, n_y, n_z)$$

とする．Aの大きさは，

$$n_2 \cos\theta_2 - n_1 \cos\theta_1$$

なので，式（3）が

$$\begin{cases} n_2 E^{'}_x = n_1 R_x - n_x(n_2\cos\theta_2 - n_1\cos\theta_1) \\ n_2 E^{'}_y = n_1 R_y - n_y(n_2\cos\theta_2 - n_1\cos\theta_1) \\ n_2 E^{'}_z = n_1 R_z - n_z(n_2\cos\theta_2 - n_1\cos\theta_1) \end{cases} \tag{4}$$

という成分で表される．屈折率の式，式（1）を$\sin\theta_2$について解いて2乗すると，

$$\sin^2\theta_2 = \frac{n_1^2}{n_2^2}\sin^2\theta_1$$

となり、さらに

$$\cos\theta_2 = \sqrt{1 - \frac{n_1^2}{n_2^2}\sin^2\theta_1} \tag{5}$$

となる．Nが単位ベクトルであることを考慮しつつ，式（5）を式（4）に代入することで，

$$\begin{cases} n_2 E^{'}_x = n_1 R_x - (\sqrt{n_2^2 - n_1^2\sin^2\theta_1} - n_1\cos\theta_1) \\ n_2 E^{'}_y = n_1 R_y - (\sqrt{n_2^2 - n_1^2\sin^2\theta_1} - n_1\cos\theta_1) \\ n_2 E^{'}_z = n_1 R_z - (\sqrt{n_2^2 - n_1^2\sin^2\theta_1} - n_1\cos\theta_1) \end{cases} \tag{6}$$

が得られる．この各式の両辺をn_2で割ると屈折光線の進行方向を表すベクトルを求められる．なお，ルートの中が負になるような角度θ_1のときは全反射となる．

2.3　空間分割レイトレーシング法

レイトレーシング法による画像生成における問題点といえば，計算時間の多さが必ずあげられる．これは，1 本の光線に対して全ての物体の交差判定を行うため，明らかに交差しない物体との無駄な交差判定が生じるためである．この問題に対しては，画面分割やバウンディングボリューム法，空間分割など様々な手法が考えだされている [1]．中でも，空間分割は交差判定の削減のみならず，それぞれの光線の計算量を均一にすることができるので，並列化を考えたときに負荷分散がある程度計れ，良い効果が得られると思われる．

図 3: 空間分割レイトレーシング

物体の存在する空間を図 4 のようにボクセルに分割し，視点から出された光線は通過したボクセルに含まれる物体のみと，交差判定する．よって明らかに交差しない物体との交差判定を行わなくてよいことになる．また，物体の存在しないボクセルは取り除かれるので明らかに物体がない領域には光線を出さなくてよい．これらから分かるように，光線は最低 1 回の交差判定を行い，最大交差判定回数も従来より大幅に減らすことができる．交差判定回数が平均化されたことで、並列処理を行う際には負荷均衡が計れるようになる．逆に，負荷均衡を計るためには，それぞれのボクセル内の物体数を一定値に近づければ良いわけである．

今回は，ボクセルに含まれる物体の数をより一定値に近付けるために，ボクセルの大きさを物体の位置により変える，という空間分割を用い，ボクセル間移動アルゴリズムには 3DDDA を用いた [6]．

3　空間分割レイトレーシング法の並列化

3.1　マルチトランスピュータシステム

トランスピュータはINMOS社が開発した通信機能を持った並列処理用マイクロプロセッサである．今回は，CPU機能として浮動小数点ユニット内蔵のT805を使用した．これはメモリのアドレス空間4GBの内，4KBの内部メモリを内蔵し高速なアクセスが可能である．他に，プロセッサ間で通信するためのリンクを4本持ち，通信速度は標準で10Mbpsであり，5Mbpsや20Mbpsにすることもできる．リンクは2本で1組，すなわち入力と出力の組合せであり，1対1の通信しか出来ない．結合網は可変で，4本のリンクで可能なネットワークなら，ファイルに記録するだけで簡単に実現できる．代表的なものとして，リング，トーラスやハイパーキューブ結合網がある．それぞれの結合網を図4に示す．

図4: 相互結合網

今回の実験で使用したシステムは，ルートトランスピュータと並列処理を行うトランスピュータ64台のトーラス結合で構成される．ルートトランスピュータはホストマシン上にあり，あるトランスピュータとしか通信できない．そのため，ルートトランスピュータが全てのトランスピュータと通信するためには，必ず決まったトランスピュータを通過する．各トランスピュータも，定義した結合網にそった通信経路しか使用できないため，他のトランスピュータを経由しなければならない．

トランスピュータ用の開発言語にはParallel Cを使用した．これは従来のC言語がトランスピュータ用の並列処理言語に拡張開発されたもので，通信，マルチスレッドの機能が拡張されている．マルチスレッドとは各トランスピュータ上の複数のタスクの中に存在する複数のスレッドのことで，これらはセマフォによる排他制御により，複数のものを並行に処理できる．例えば，通信は完全に同期がとられ，送り側と受け側の意志が一致したところで行われる．一致するまでは，送り側や受け側は処理を止められて待ち状態になる．そこで通信と，本来すべき処理をそれぞれスレッドとして処理を行い，排他制御を用いることで並行に処理すれば，通信が処理に及ぼす影響を減らすことができる．

他の並列処理言語としてはOCCAMがある．これはCSP(Communicating Sequential Processes)を，より汎用的な使いやすい言語として，継承したものである．逐次処理を行っている複数のプロセスが，互いに影響しあいながら処理をすすめる形の言語である．通信はParallel Cと同じように同期がとられ，通信を行うプロセッサがそれぞれ，受信，送信状態になった時に行われる．性能的には，基本的にParallel Cより優れているが，細部まで比較すると，Parallel Cが優れている点もあり，またC言語に近い点から馴染みやすいという利点もあげられる[7][8][9]．

3.2 透過物体を含むシーンの並列計算

実験に使用したシーンは，拡散反射，鏡面反射，透過物体を含むものである．鏡面反射や透過による光線は，拡散反射のものと同様に扱えるため，本研究室で構築されたシステムと同様の流れで処理できる．すなわち，トランスピュータ1台を64個に分割したボクセルそれぞれに割り当て，光線のボクセル間の移動をトランスピュータ間の通信に置き換えることで，実現できる．

並列処理手順としては，ボクセルや物体の情報を対応する各トランスピュータに分散させて送信し，ルートトランスピュータが画面分割（インタリーブ分割）により出す光線の情報を，通過するボクセルのトランスピュータに送信することで計算が開始される．光線が鏡面や透過物体と交差した時に光線の数が増えるが，増加した光線も通信により，光線の進行先のボクセルを担当するトランスピュータに情報を提供することで，進行先のトランスピュータが追跡する．各プロセッサは1つでも輝度が得られた時点で，ルートに輝度や他の情報を送信し，さらに光線追跡を続ける．それぞれの光線から得られる輝度はルートが光線ごとに合計し，最終的な輝度を得る．各光線の追跡の終了は，ルートが各プロセッサから送られてくる情報をもとに決定する．透過物体が表現できるようになったことで，各光線の計算量により大きな違いができるが，処理を終えた光線があれば，ルートはすぐに次の光線を進行先のプロセッサに割り当てることで解決される．ルートが各プロセッサで光線追跡を終了させることを決定する要素としては，重みの情報を用いる（3.3節）．また，

光線を出す順序は画面分割のインターリーブ分割を用いて，スクリーン全体に拡散するように出していく．

各プロセッサの光線追跡は，物体の性質判定から始まり，透過物体，鏡面物体，拡散反射物体の順に追跡を行う．ルートからの追跡終了以外にも，あらかじめ設定された透過，反射最大回数 (MAXLEV) 以上の追跡は行わずに終了する．この場合はプロセッサ自身で追跡を終了し，ルートから次の光線情報を受信し，処理を続ける．設定する最大透過，反射回数はプログラム中で定数として設定するため，設定した回数以上の透過，反射が起こる画像においては，打ち切られた状態で画像が表示される．

今回の実験で用いたトーラス結合では，ほとんどの通信の時にほかのプロセッサを通過しなければならない．この場合，通過するプロセッサの処理を停止させることなく通信を行えるように，通信プロセスとレイトレーシング計算プロセスをマルチスレッドで実現し，通信プロセスがレイトレーシング計算部と競合しないように，セマフォにより管理している [6]．空間分割レイトレーシング法を並列化した時の流れを図 5 に示す．

図 5: 並列計算

3.3 リターンパケット

リターンパケットとは，各スレッドの光線追跡により得られた物体の輝度を，表示するスクリーン座標とともに，ルートトランスピュータへ届ける役割を持ったパケットである．加えてこのパケットは，ある光線についての追跡の終了をホストに知らせる“重み”というデータを有している．この重みにより，ホスト側は反射，透過光線の追跡の終了を認識し，次の光線の追跡を始める．

重みは，光の減衰を簡易的に表すものでもあり，プログラム中で設定してある最大透過，反射回数とは別のものである．光は，透過，反射していくごとに弱いものになり，最終的には物体の輝度を得られない程度の明るさまで減衰する．よって，設定した最大透過，反射回数より少ない透過，反射回数の場合でも，物体の輝度を得ない場合がある．実現方法としては，タスク側では重みの初期値は 1，ホスト側では合計 1 の重みを受信すれば追跡終了とし，以下の制約によりタスクで制御される．

- 拡散反射物体と交差⇒ 重みはそのまま
- 反射，透過のどちらか一方の要素をもつ物体と交差⇒ 重みを 1/2
- 反射，透過の両要素をもつ物体と交差⇒ 重みを 1/4

図 6: リターンパケットの重み

リターンパケットは，光線が物体と交差してから，透過，反射などで次の物体と交差判定を行う前に送信される．反射，透過の両要素をもつ物体と交差したとき，反射と透過のリターンパケットの送信する輝度は同じものとなるので，反射側については輝度を送信しない．しかし重みについてはその光線の追跡の終了判定に使用されるので，透過光線のリターンパケットの送信において，反射光線分の重みも送信する．

4　実験と考察

4.1　実験方法

実験は，並列計算機トランスピュータ64台上で，トーラス結合網での実験を行った．6つの画像(jacks，skull，teapot，tetra，mount，balls)について1，4，16，64台のトランスピュータで実行時間を計測した．またその中で最高の並列性を示した台数において，同時に飛ばす光線の数が1，8，16，32，64，96，128本の時の実行時間を計測することで，透過物体を含むシーンにおける適切な光線本数を見いだした．

4.2　実験結果と考察

4.2.1　プロセッサ台数による並列効果

透過物体を含む6つの画像において，物体数，トランスピュータを64台使用したときの実行時間を表1に示す．

表1 :台数による時間変化 (秒)

	物体数	64台
jacks	312	224
skull	1243	352
teapot	2329	1603
tetra	4096	766
mount	8192	3065
balls	14726	3978

これは，空間分割レイトレーシング法を並列化した際に，どれだけの速度向上がみられたかを示すものである．残念ながら，64台以外のプロセッサでの実験ができなかった．実験においては，32，64台に優れた並列効果が得られると思われる．

4.2.2 1次レイの本数による時間変化

同様の6つの画像について，4.2.1から最も優れた並列効果が得られた台数(64台)において，物体数，1次レイの本数を1，8，16，32，64，96本に変化させたときの実行時間を表2と，図7に示す．

表2 :1次レイの本数による時間変化(秒)

(64台)	物体数	1本	8本	16本	32本	64本	96本
jacks	312	2957	452	277	221	224	230
skull	1243	4978	694	437	372	352	364
teapot	2329	20874	2990	1845	1555	1603	1663
tetra	4096	14066	1781	1060	818	766	779
mount	8192	40772	5721	3419	3028	3065	3099
balls	14762	49622	7276	4331	3975	3978	3997

図7: 1次レイの本数による時間変化

1次レイの本数を変えるということは，同時に処理するスレッドの数を変えるということに等しい．よって，32本のレイを出すということは，32個のスレッドをはしらせるということで，実験では64台のプロセッサを使用してはいるが，実際に稼働しているプロセッサは32台ということになり，飽和した状態になっていない．また，96本以上の時は，プロセッサ台数以上のスレッドが動くということであるが，1つのプロセッサで複数のスレッドがはしるのではなく，64以上のスレッドは生成されるが，待ち状態となってしまい，並列に処理されない．

結果は，jacks，teapot，mount，balls では 32 本，skull，tetra では 64 本の光線を出すのが時間的に 1 番優れていると分かった．より詳しい考察のためには，全てのプロセッサの稼働率や負荷分散を調べて，理論からデータを予想していくことも，大切だと思われる．

4.2.3 1 次レイの本数に関する透過物体を含まないシーンとの比較

6 つ全ての画像において，物体数，4.2.2 で得られたデータと，透過物体を含まないシーンのデータを図 8，9，10 にグラフで比較した．mirror のついたグラフは透過を含まない，鏡面反射までを含むシーンのグラフを表す．表 3 に透過物体を含まないシーンにおいて，1 次レイを変化させた時の実験結果を示す．

表 3 :1 次レイの本数による時間変化 (秒)

(64 台)	物体数	8 本	16 本	32 本	64 本	96 本
jacks	312	410	253	199	203	211
skull	1243	562	350	306	301	312
teapot	2329	2408	1481	1104	1117	1163
tetra	4096	1428	835	588	570	578
mount	8192	4441	2682	2440	2459	2475
balls	14762	7426	4810	4604	4627	4649

図 8: 1 次レイの本数による時間変化の比較 (jacks，skull)

図 9: 1 次レイの本数による時間変化の比較 (teapot，tetra)

結果は，時間的には今回のデータの方が上回っているが，適切な本数としては同じ結果が得られた．ただ，balls の画像における実行時間が，透過物体が含まれることで減少している．データはもちろん，視点などあらゆる点で同じデータを用いたにもかかわらず，全ての本数で減少している．balls の拡散反射物体の実行時間については，32 本において 2566 秒で，ほかの本数でも予想通り，透過性質が加わることで時間増加が見られる．考えられるのは，透過物体を含まないシーンで，各々の光線で計算量にばらつきがあったものが，透

図 10: 1 次レイの本数による時間変化の比較 (mount，balls)

過の性質が加わることで著しく平均化されてしまった，ということである．すなわち，光線の分岐増加による計算量の増加よりも，各々の光線の計算量の平均化による負荷均衡が，大きな影響を与えたということである．追跡する光線が増えたのだから，実行時間はかかって当然と思っていたが，物体数が多くなることで，物体空間をいつまでも反射し続ける光線と，1 度反射するだけで物体空間を飛び出してしまう光線とでは，計算量に大きな違いがある．ところが，透過が含まれることで，1 度しか反射しなかった光線が，物体の大変集中した辺りに透過していったり，物体中に透過してから，物体の内側で反射を繰り返したとすれば，計算量が平均化されることも考えられる．あくまで，予想の域での仮説であり，負荷均衡によりこれほどの実行時間の減少につながるのか疑問であるが，各々の光線の負荷分散を調べることで確認できると思われる．他の画像でも，balls と同じ現象が起これば実行時間が減少するかもしれないが，skull，tetra，mount では，3 角形で構成された画像で，1 度透過しても次に反射する物体が，目で確認できない後ろ側にないので，各々の光線が平均化されることは少ない．jacks については，物体の存在する範囲や数が少ないため，平均化されにくい．teapot では床を表す物体が影響しているのではないかと思われる．

時間的に上回るのは，透過物体がシーンに含まれるようになり，光線追跡に時間がかかるようになったので，当然の結果だと思われる．平均的には 1.1∼1.4 倍の時間がかかっている．balls においては 0.86 倍に減少している．

1 次レイの本数は，同じ本数が適切なものとなったが，これは物体データの全てを透過鏡面物体に変更して，実験を行ったからであると思われる．1 次レイの本数というのは，各々の光線の負荷均衡にも関係している．これは画像にも大きく左右され，全ての物体が透過

性質を付加された場合では，画像全体としてのバランスがあまり変わらないので，光線本数による変化率もあまり変わらなかったと思われる．

4.3 生成画像

今回の実験に用いた6つの画像を図11，12，13に示す．

図11: jacksとskull

左の画像(jacks)は，3次元的に組み合わされた十字架で，十字の棒を通して後ろの棒が見えるのがわかる．右の画像(skull)は，頭蓋骨の後頭部の部分が存在しないが，左側面の部分が左目のあたりを通して確認できる．

図12: teapotとtetra

左の画像(teapot)は，ティーポットがチェックの床の上におかれている様子を表している．ティーポット自体が透過物体で，床は拡散反射物体である．ティーポットを通して後ろの床が確認できる．右の画像(tetra)は，3角垂が中心に向かって積み上げられている様子である．この画像からは透過物体を含んでいるシーンだと確認しにくい．

図 13: mount と balls

左の画像 (mount) は，山を表しているが，後ろに物体が存在しないために，確認しにくいと思われる．右の画像 (balls) は，大小の球が無造作に配置されていて，全ての球が透過の性質を持っている．左上の球で透過，右上の球で反射が確認できる．

5　おわりに

今回の論文では，拡散反射物体と鏡面物体のみを対象とした，空間分割レイトレーシング法の並列化の研究に，さらに透過物体を加えたシーンについての並列化について述べた．視点から出された光線が，物体表面で2本の光線に分離していくことによる並列性能への影響や，適切な1次レイの本数の検討を以前のデータと比較して記した．実験結果から，透過物体を含むシーンの並列性を確認できた．空間分割は，光線がボクセルのどの方向から進入してきても効果を発揮するので，反射や透過光線に対しても大きな効果が得られた．逆に，空間分割レイトレーシング法が並列化の際に，負荷分散にどの程度の影響を与えるのかについて実験できなかったことが残念である．これは，適切な1次レイの本数を考えていく上でも重要であるし，4.2.3で述べたballsへの仮説の確認にもなると思われる．また，プロセッサの台数を変化させることで，並列効果を確認することもできなかった．今までの様々なトランスピュータ上での実験や，本研究で扱う画像データの量から，64台での処理が最大の並列効果が得られるとの判断で，1次レイの実験を行ったが，1次レイの実験から，32台が稼働している場合に優れた結果を出す画像もあったので，実験の価値はあると思われる．

今後の研究として，さらに広範囲の性質を含むシーンに対しての，空間分割レイトレーシング法の並列性能の実験があげられる．そのために，現在は球と三角形だけの物体登録を拡張することも求められる．また，3DDDA計算時間とレイトレーシング計算時間を考慮した，適切なボクセル数の検討や，ボクセルデータ作成過程の並列化があげられる．

謝辞

本研究の機会を与えてくださり，なおかつ数々の御助言を頂きました，山崎勝弘教授に深く感謝の意を表します．また，これまで数々の御助言を頂きました研究室の皆様に感謝致します．

さらに，大変お忙しい中，様々な質問にお答えいただいた上嶋明様，吉谷崇史様，色々とアドバイスしてくださった浅川達弥先輩に，深く感謝致します．

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,pp.9-41,1991.

[2] 小堀研一, 春日久美子,”基礎から学ぶ図形処理”, 工業調査会,pp.52-103,1996.

[3] 鷺島敬之, 西澤貞次, 浅原重夫,”並列図形処理”, コロナ社,pp.1-7,pp.97-102,1991.

[4] 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学科卒業論文,1995.

[5] 吉谷崇史,”空間分割による並列レイトレーシング法”, 立命館大学大学院理工学研究科修士論文,1997.

[6] 吉谷崇史,”適応空間分割による並列レイトレーシング法”, 情報処理学会第５４回全国大会論文集,4Q-6,1997.

[7] 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,pp.159-215,1990.

[8] 木村博昭, 福島みのる,”トランスピュータによる並列処理”, 海文堂,pp.1-9,pp.171-175,1990.

[9] ”TRANSPUTER APPLICATIONS NOTEBOOK Architecture and Software first edition”, INMOS,pp.41-42,pp.145-146,1989.