グラフィクスとCAD 53－4
（1991. 10. 23）

動的部分木を用いた光線追跡法

成瀬 正　　新谷 幹夫　　斎藤 隆文
NTTヒューマンインタフェース研究所

光線追跡法の高速化の本質は、各画素に対して、その画素に投影される物体を効率よく求めることにある。その方法として本論文では、階層化された木構造データを前提とし、それから作られる部分木を用いた手法を提案する。すなわち、画像の小領域に投影される物体からなる部分木を、与えられた木構造データから動的に構成し、小領域内の画像生成は部分木を用いて行う手法である。提案手法ではさらに、部分木に対して木のノードレベルで視点からの距離によるソーティングを行うとともに、小領域内の一様性テストを併用することで、より一層の高速処理を実現する。このアルゴリズムをインプリメントして実験を行った結果、部分木を用いることにより、交差判定回数を著しく削減でき、部分木を動的に作成するコストも小さいことがわかった。また、従来高速であるといわれているArvoら[2]の手法と比較しても本手法はより高速であることがわかった。

RAY TRACING USING DYNAMIC SUBTREE

Tadashi Naruse　　Mikio Shinya　　Takafumi Saito
NTT Human Interface Laboratories

In ray tracing, it is essential to efficiently obtain an object projected on each pixel of the screen. We propose an efficient algorithm that uses tree structured data to obtain the object. The key idea of this algorithm is the use of a subtree. From a given tree, we dynamically construct a subtree consisting of objects that are projected on a subregion of the screen. Then we use this subtree for tracing rays in the subregion. We also use a sorting technique together with uniformity testing of the subregion. Experimental results show that our algorithm can greately decrease the number of intersection computations and that the cost of constructing a subtree is low. These experiments show that our algorithm is faster than Arvo's algorithm.

1. まえがき

T.Whitted [1] により提案され、脚光を浴びた光線追跡法は、幾何光学に基づいて光学系を忠実にシミュレートする手法であり、極めて写実的な画像が生成できるという特長を持つが、一方で、その計算にかなりの時間を要するという問題点を有する。この計算時間の大半は、光線と物体の交差判定に費やされる[1]。

光線追跡アルゴリズムの高速化の本質は、つきつめれば、各画素に対して、その画素に投影される物体を効率よく求めることにある。それを目的として従来より種々の高速化技法が提案されてきた。従来の手法を分類すると、①データの階層化とBounding Volume（ＢＶ）の使用、②Ray coherencyの利用、に大きく分けられる。

データの階層化とＢＶの使用により交差判定回数の削減を図る方法がよく知られている。OctreeやConstructive Solid Geometry（ＣＳＧ）モデルを用いた階層化はその一例である。これらのモデルでは階層化された木の各ノードに対し、そのノードの子孫となる葉ノードのプリミティブをすべて包含するようなＢＶを必要に応じて設定する。これらのＢＶに対して交差判定を行うことにより交点計算の回数を大幅に削減できる。　一方、ある光線とその近傍の光線とは似た振る舞いをすることが多い。このような Ray Coherency をうまく利用すれば光線追跡法を高速化できることが知られている。この代表例として、Arvoら[2]の方法があげられる。Arvoらの方法では、類似した光線をグルーピングし、これらと交差する物体の候補を求めることにより、交差判定計算を削減しており、高速化効果は大きい。

本論文では、データの階層化とＢＶを利用した、より高速な光線追跡アルゴリズムを提案する。この手法では、まず木構造（２進木）のＢＶを構築しておき、光線追跡は画像をいくつかの小領域に分割して処理する。個々の小領域ごとに、木構造ＢＶを調べ、小領域に投影されるＢＶからなる部分木を動的に作成する。この部分木を用いて、領域内の画像生成を行う。一般に、部分木はもとの木に比べて階層が浅いため、ＢＶとの交差判定回数は大幅に削減される。本手法ではさらに高速化を図るため、①部分木の各ノードレベルでの視点からの距離によるソーティングを行う、②小領域内が同一の物体の投影像か否かをテストする、という技法も組み合わせる。これらの相乗効果により、本手法はArvoらの手法よりも高速に画像生成できることが、実験結果から示される。

2. 光線追跡法の高速化技法

本節では、画素に投影される物体を効率よく求めるという観点から光線追跡法の種々の高速化技法を考察し、インプリメントに際して考慮すべき点を検討する。以下では、視点から追跡する光線を１次光線、反射・屈折光線を２次光線と呼ぶ。

２．１　物体定義データの階層化

物体定義データから木構造のＢＶを構築する場合、一般に以下の手順で行う：

①まず、形状定義の最小単位（プリミティブ）を階層的にクラスタリングして木構造とする。

②次に、木の各ノードごとに、その下のすべてのプリミティブを包含するＢＶを設定する。

光線追跡の効率を左右する要因として、①ではクラスタリングの方法および各ノードの枝（子ノード）の数が、②ではＢＶの形状が、それぞれあげられる。ここでは、このうちの子ノードの数とＢＶ形状について考察する。

２．１．１　　子ノードの数と交差判定回数

各ノードにおける子ノードの数と、交差判定回数との関係を考察する。例として、完全なｋ進木で階層化した場合を考える。また、木は平衡しているとする。いま、「条件１：ある光線があるノードのＢＶと交差するならば、そのｋ個の子ノードのＢＶの中でただ１個と交差する」という状況を考える。これがすべての交差ＢＶに対して成り立つとすれば、その光線に対する交差判定回数Ｎは、

$$N = k \cdot \log_k n$$

となる。ここで、ｎはプリミティブ数である。

ｎを固定したときＮを最小化するｋは、ｋ＝ｅ（自

然対数の底）である。ｋは整数であるから実際にはｋ＝３がＮの最小値を与える。ついで、ｋ＝２，４となる。２と４は同じ値を与える。したがって、交差するＢＶに対して平均１個程度の子ノードのＢＶと交差するような条件下では、２～４進木による階層化が有利となる。

ｋ個の子ノードのＢＶのうち複数のＢＶと光線が交差すると条件１は成立しない。しかしながら、交差する複数のＢＶから何らかの方法で視点に最も近いプリミティブを含むＢＶを一意に選択できれば、この条件は成立する。本論文では、この条件が多くの光線に対して成立するようにアルゴリズムを構築していく。それゆえに本論文では、クラスタリング処理の単純さや２．２．２節で述べるソーティングの効率化を考え、２進木を採用する。

２．１．２　ＢＶの形状と交差判定コスト

ＢＶはできるだけ体積が小さくなるように設定することが望ましいが、交差判定コストを考慮した設定をしなければならない。いま、図１(a)に示すような各ノードにＢＶが設定された木構造を考え、その画面（スクリーン）への投影が図１(b)となったとする。また、ノードn1の投影面積をＳ1とする。ノードn1に光線が当たったときに限り、その子ノードのＢＶとの交差判定が行われる。したがって、その子ノードとの交差判定に要する時間の、画面全体での総和Ｔは、

$$T = k \cdot C \cdot S_1$$

となる。ここで、ｋは子ノードの数（この例では２）、Ｃは子ノードのＢＶ１個との交差判定コストである。

条件１が成立するならば、サーチする木の各ノードに対してこの式が成立する。したがって、単純な形状のＢＶを用いても交差判定コストが小さければ、トータルの計算時間は少なくなり得る。（投影面積が２倍になってもコストが半分なら計算時間は同じである。）

以下では、投影面積は大きくなるが判定コストの小さな、軸（世界座標）に平行な辺を持つ直方体を採用する。

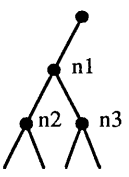
２．２　　１次光線に対する高速化

２．２．１　動的部分木

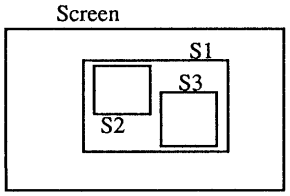
画面の小領域を考えた場合、そこに投影されるプリミティブの数は限られている。そこで、小領域毎に与えられた木から投影されるＢＶを抽出し、動的に部分木を作る。小領域中では部分木を利用して光線追跡することにより、高速化を図ることができる。本論文では、画面をメッシュ状の小領域に分割することを考える。このとき、部分木は、視点と領域の４隅の点から作られる角錐に入るかあるいは交差するＢＶから作られる。このとき、図２に示す最適化も行う。この部分木作成にさほどコストがかからないことは次の考察からわかる。

（１）角錐の内外判定

角錐は４つの平面から構成されるが、前述の小領域分割を行う場合、これらの平面とＢＶとの位置関係を小領域毎に求める必要はない。画面の大きさをｎ×ｎ画素、小領域の大きさをｍ×ｍ画素とすれば、角錐を構成する平面の数は全体で２（ｎ／ｍ＋１）だけである。したがって、これらの平面について、各ＢＶがどちら側にあるか（もしくは交差するか）を計算すれば、角錐に対する内外判定は容易にできる。また、ＢＶが階層化されていると、あるＢＶに対し、そのＢＶが平面のどちらか一方の側に属すれば（すなわち、面と交差しなければ）、そのＢＶの下のＢＶに対する交差判



(a)



(b)

図１　ＢＶの投影

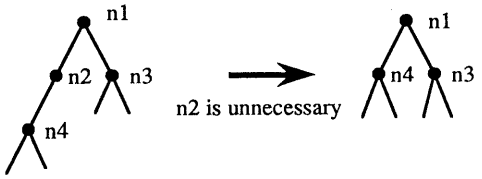


図２　部分木の最適化

定は不要となる。この事実を使って、角錐と交差するか含まれるプリミティブからなる部分木を次のようにして作ることができる。

（２）動的部分木作成

木の各ノードに４つのフラグを与える。各フラグは、角錐の面に対応し、ノードに対応するＢＶあるいはプリミティブが面の内側にあるか、外側にあるか、あるいは交差するかを示す。面に対して木のルートノードからはじめて各ノードのフラグを設定していく。このときあるノードで面の内側かあるいは外側になったらその子ノード以下の当該面に対するフラグ設定はしない。

小領域の画像生成をたとえばスキャンライン方向に行うなら、角錐を構成するスキャンライン方向の２つの面は、その方向の各角錐に共通である。したがって、これらの面に対するフラグはスキャンライン方向毎に１回設定するだけでよい。他の２面に対するフラグ設定は小領域毎に行うが、それはスキャンライン方向の２面に挟まれるＢＶに対してのみ行えばよい。

このようしてフラグが設定されたら、そのフラグを見ながら部分木を構成する。その際、面の外側あるいは内側と設定されているフラグは、その情報を子ノードに継承していく。

２．２．２　ソーティング

より視点に近いプリミティブが画像面に投影され、そのプリミティブを包含するＢＶは他のプリミティブを包含するＢＶと比べてもより視点に近いであろう、という考えのもとに、部分木の各レベルで視点からの距離によるソーティングを行う。すなわち、あるノードのＢＶと光線が交差し、その子ノードに対して交差判定を行う必要が生じたら、子ノードを視点からの距離の近い順に並べ替える。これは、光線毎に行う。

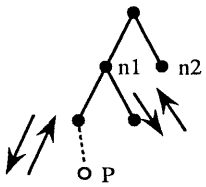
深さ方向優先の木探索を行う場合、木の各レベルでソーティングが行ってあれば、図３に示すように、ある子ノード以下を探索して得られたプリミティブの視点からの距離が、次の子ノードに設定されたＢＶの視点からの距離より近いとき、その子ノード以下の探索は不要となる。このようにして、ソーティングを利用した枝刈ができる。

２進木を用いれば、子ノードのソーティングは１回の比較だけで済み、極めて効率がよい。

２．２．３　領域内の一様性テスト

小領域内が同一のプリミティブの投影像であるとき、この領域内は一様であると呼ぶ。領域内が一様であることが何らかの方法でわかれば、この領域では、そのプリミティブのレンダリングを行えばよい。このことは、領域内で木の探索が不要となることを意味し、高速化に与える効果は大きい。ただし、このテストは、できる限り単純なものであることが要求される。

ここでは、次のような簡単なテスト法を提案する。小領域の４隅の点に対して光線追跡する。この段階で、部分木は、４番目に追跡した光線でソーティングされている。今、４隅の点のプリミティブが同一であったとする。それをＰとする。このとき、小領域内でプリミティブＰより視点に近いところに他のプリミティブが無いことを調べればよい。部分木をLeft-most Depth-Firstでサーチし、プリミティブＱを見つける。Ｑは木の一番左端にある。木の構成のしかたから、ＱはＰより視点に近い。ＰとＱが同一でなければこの領域は一様でないとする。いま、ＰとＱが同一であったとする。このときＰの弟ノードを調べる。弟ノードはプリミティブかＢＶが設定されているかいずれかである。それをＲとする。Ｒが４番目に追跡した光線と交差するか調べる。これは、フラグをみるだけでよい。



1. Searching descendants of node n1, get a primitive P closest to eye. Assume that distance is t1.
2. Assume that distance between BV set at n2 and eye is t2.
3. If t1<t2, it is not necessary to search descendants of n2.

図３　ソーティングを利用した枝刈

交差すれば、領域内は一様である。そうでなければ、一様でないとする。

このテストは領域内が一様であっても、一様でないという判定を与える場合があるが、領域内が一様と判定されたときには、プリミティブPの前に他のプリミティブが無いことは保証される。

領域内が一様でないときは、領域を2分割して一様性テストをする。領域の大きさが閾値を下回らない範囲でこれを再帰的に繰り返す。

2．3　2次光線（反射・屈折光束）に対する高速化

画面の矩形領域を通過する光束は角錐台で表現することが可能であり、2．2節までに述べてきた手法が適用できる。しかし、一般の反射・屈折光束を正確に記述することは困難である。そこで、本論文では、

①反射・屈折光束を近似し、それを含む近似的な光束のバウンディングボリューム（以下、ＰＢＶ(Pencil Bounding Volume)と呼ぶ）を求める。

②近似的なＰＢＶに対し、部分木を求め、一様性のテストを行う。

③各反射・屈折光線に対し、①で求めた近似ＰＢＶに含まれるか否かを調べ、含まれる場合には②で得られた結果を利用する。

というアプローチをとる。ここで問題となるのは、いかにして近似ＰＢＶを求めるかという点である。

光束の拡りが小さい場合には、近軸近似が反射・屈折の近似として有効であることが知られている[8]。そこで本論文では、近軸近似を利用して近似ＰＢＶを求めることにする。

近軸理論によれば、一点（たとえば視点）を発した近軸光束は2つの焦線を持ち、その焦線の方向は直交する[8]（脚注）。すなわち、近軸光束は図4に示すように、互いに直交する交線を有する2つの平面の組に囲まれた領域として記述される。したがって、この領域を光束の近似として用いることは合理的である。しかし、実際の光束は理想的な近軸光束ではなく、多くの光線が近軸光束の領域外にはみ出すことになる（図5(a)）。そこで、光束の4隅の光線を基に、ＰＢＶを

（脚注）直感的には、光束の波面が2次曲面で近似されることに対応する。

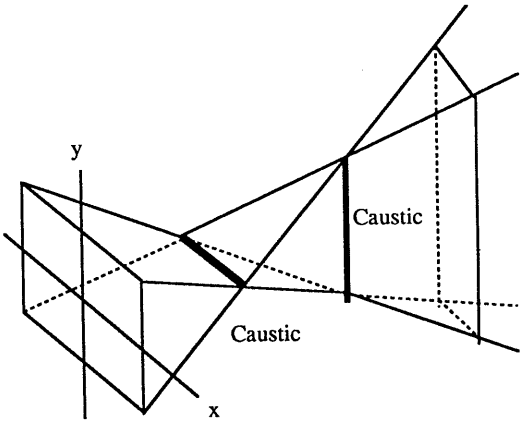
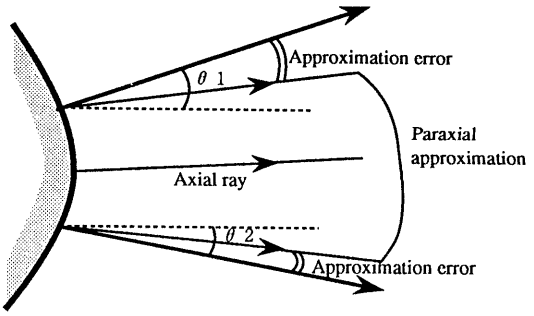
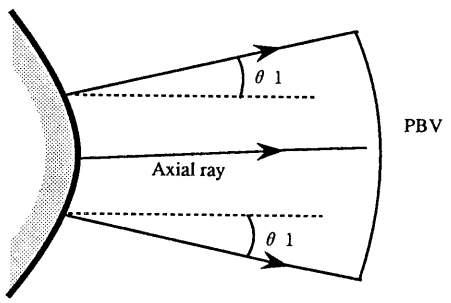


図4　近軸光束の一例



(a)



(b)

図5　ＰＢＶの構成法

求めることを考える。この求め方は多数存在するが、簡単な方法として、４隅の光線の振れ角のうち最も大きな角をＰＢＶの拡り角とする手法がある。図５(a)に示した２次元の例では、$|\theta 1| > |\theta 2|$であるので、図５(b)のようなＰＢＶが得られる。具体的には、以下の手順で近似ＰＢＶが求められる。

①画面小領域の４隅に対応する光線の反射・屈折光線ｉをSnell則より求める。

②得られた４本の光線の平均をとり、軸光線とする。

③軸光線をｚ－軸とする適当な光線座標系ｘ－ｙ－ｚを求める。ただし原点は、４本の光線の始点の内、いちばん手前のものがｚ＝０となるようにとる。

④４本の光線の始点および方向を光線座標系に変換し、ｚ＝０の面との交点$r_i=(r_{xi}, r_{yi})$、および方向$s_i=(s_{xi}, s_{yi})=((dx/dz)_i, (dy/dz)_i)$を求める。

⑤$r_{xmax}=\max(r_{xi})$, $r_{xmin}=\min(r_{xi})$, $r_{ymax}=\max(r_{yi})$, $r_{ymin}=\min(r_{yi})$および$s_{xmax}=\max(|s_{xi}|)$, $s_{ymax}=\max(|s_{yi}|)$を求める。

⑥求める４平面は次式で与えられる。

$$x - s_{xmax} * z = r_{xmax}$$

$$x + s_{xmax} * z = r_{xmin}$$

$$y - s_{ymax} * z = r_{ymax}$$

$$y + s_{ymax} * z = r_{ymin}$$

なお、ｘ，ｙ軸のとり方は任意であるが、焦線方向にあわせてとるのが効率的である。

この近似ＰＢＶ作成法は最も単純なものであり、さまざまな変形が考えられる。

近似ＰＢＶ作成の最適化は今後の課題として残されており、処理効率の実験的な解析が必要であると考えられる。

２．４　光線追跡アルゴリズム

上で述べた高速化技法を用いて光線追跡アルゴリズムをインプリメントできる。図６に概略アルゴリズムを示す。

３．高速化効果の評価

前節で述べた高速化技法のうち、１次光線に関してその効果を調べる。そのために、以下のアルゴリズムを用意した。

Ａ１　：ＢＶのみを使う。

Ａ２　：ＢＶとソーティングを使う。

Ａ２’：部分木とＢＶを使う。

Ａ３　：部分木とＢＶおよびソーティングを使う。

Ａ４　：部分木、ＢＶ、ソーティングおよび一様性テストを使う。

Ｂ１　：Arvoらのアルゴリズム

Ａ１～Ａ４は２．２節で提案した各技法の組み合わせを変えてインプリメントしたものである。また、Ｂ１は比較の対象としてArvoらのアルゴリズムの主要部を１次光線に対してインプリメントしたものである。手法の効率を論じるには、Ａ３とＢ１の比較が公平である。

３．１　実験に用いたデータ

実験には、次の物体定義データを使用した。

１）空間内に一定の半径の球を規則的に並べる。２通りの半径の球を用意する。

２）空間内にランダムな半径を持つ球をランダムに配置する。

３）１個および８個のティーポット

１）と２）はアルゴリズムの基本特性を調べるため、３）は一般の画像に対する特性を調べるために用意したデータである。画像生成例を写真１～５に示す。生成した画像の画素数は、512×512である。また、小領域は8×8の大きさを基本とした。写真１～３のプリミティブ（球）の総数は512であり、写真４、写真５のプリミティブ（三角形パッチ）は、それぞれ552、4416である。ＢＶは軸に平行な直方体を使用した。実験に用いた計算機はsun4/260である。

３．２　実験結果

（１）画像生成実験

それぞれのアルゴリズムを用いて、写真に示した画像を生成し、生成時間、交差判定回数を調べた。結果を表１に示す。この結果から次の点が指摘できる。

・表１(a)からわかるように、いずれの場合もアルゴリズムＡ４が最も短時間で画像を生成する。また、Ａ３とＢ１を比較しても、Ａ３の画像生成時間がＢ１より少ない。すなわち、提案アルゴリズムはArvoらのアルゴリズムと同等かそれ以上高速であることがわかる。

・Ａ１とＡ２’の比較から部分木が高速化に寄与す

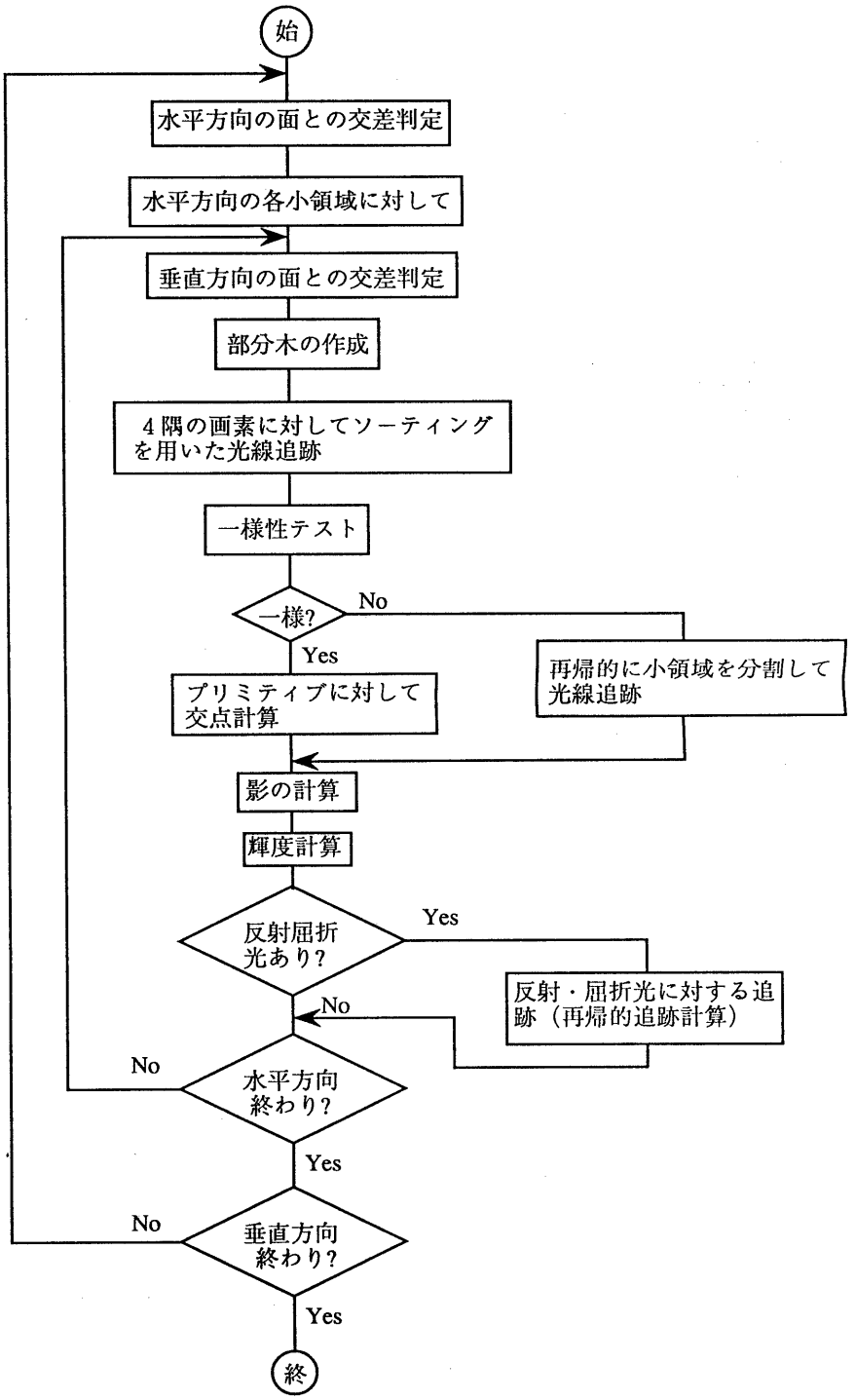


図6　提案アルゴリズムの概略フロー

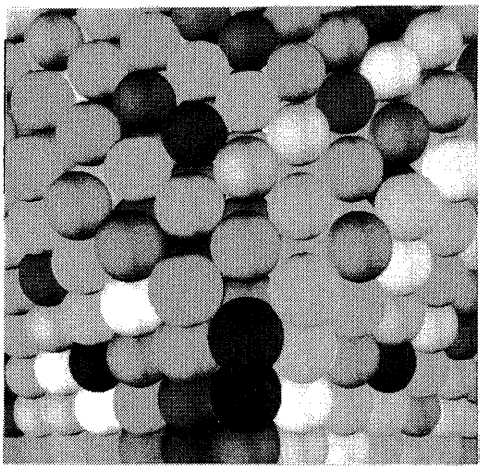


写真　1

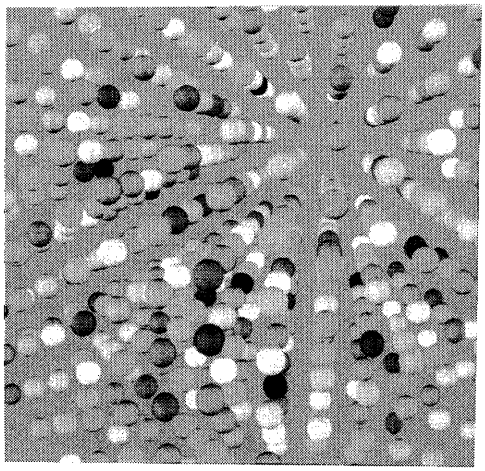


写真　2

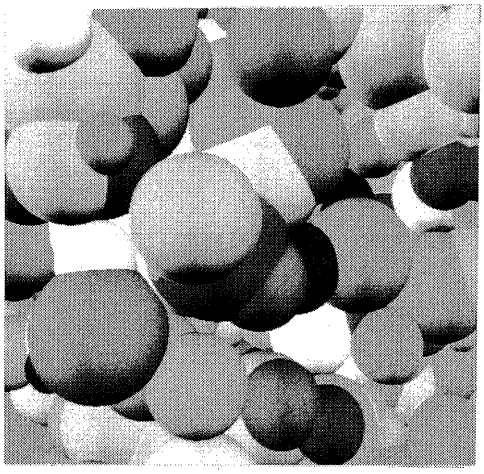


写真　3

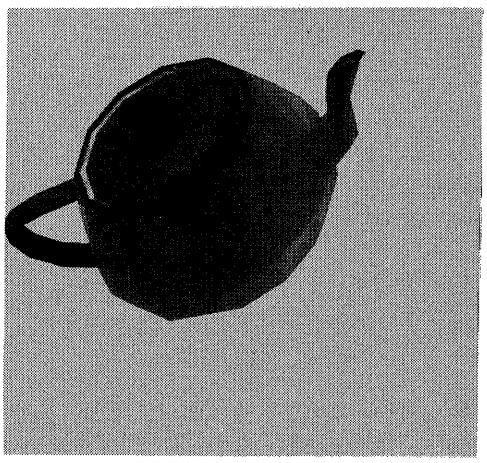


写真　4

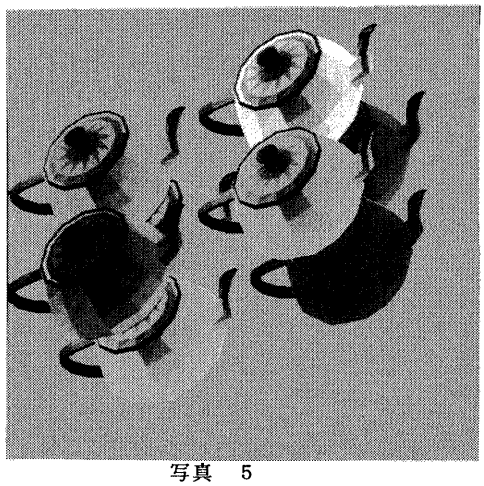


写真　5

表1　光線追跡法の計算時間と交差判定回数
（a）計算時間

ｱﾙｺﾞﾘｽﾞﾑ	写真1	写真2	写真3	写真4	写真5
A1	36'50"	20'46"	55'18"	12'19"	29'08"
A2	14'07"	17'52"	21'18"	8'42"	18'52"
A2'	5'12"	2'13"	7'16"	3'08"	7'56"
A3	4'39"	2'17"	6'46"	2'29"	5'19"
A4	3'45"	2'10"	5'22"	2'16"	5'03"
B1	6'55"	2'51"	9'34"	2'29"	6'06"

表2　小領域の大きさ（画素数）と計算時間（アルゴリズム：A4）

小領域の大きさ	写真1	写真2	写真3	写真4	写真5
4×4	5'29"	2'57"	7'23"	2'34"	5'27"
8×8	3'45"	2'10"	5'22"	2'16"	5'03"
16×16	3'45"	2'35"	5'17"	2'39"	6'30"
32×32	4'39"	4'05"	5'57"	3'30"	9'13"

表1　光線追跡法の計算時間と交差判定回数（続き）
（b）交差判定回数

アルゴリズム		写真1	写真2	写真3	写真4	写真5
A1	BV	57.9	37.9	86.6	20.4	50.9
	prim	19.8	6.88	28.8	4.61	9.60
	総数	20355	11744	30245	6552	15849
A2	BV	20.2	29.7	30.9	13.5	30.5
	prim	3.75	5.01	7.28	2.53	4.69
	総数	6281	9087	10008	4190	9228
A2'	BV	3.72	0.66	5.68	2.56	8.12
	prim	3.38	1.61	4.39	2.78	7.04
	総数	1859	595	2640	1400	3974
A3	BV	3.16	0.65	5.09	1.86	4.88
	prim	2.27	1.56	3.48	1.60	3.42
	総数	1424	581	2248	908	2174
A4	BV	1.93	0.65	3.18	1.75	4.61
	prim	2.04	1.79	2.99	1.52	3.20
	総数	1029	546	1608	893	2115
B1	prim	7.80	3.10	10.8	2.51	5.79
	総数	2045	813	2829	657	1516

BVとprimの項は光線1本あたりの交差判定回数（全光線に対する平均）
総数の項は画像面全体での総交差判定回数（BV＋プリミティブ）（$\times 10^3$）

る効果が極めて高いことがわかる。このことは、表1(b)に示されるように、A2'の交差判定回数が著しく減少することから得られる効果である。

・領域の一様性テストの効果は、プリミティブの大きさに依存する。写真2のように一様領域の少ない場合でも、実験結果は、一様性テストのオーバヘッドを加味しても、高速化効果があることを示している。

・写真2の例では、A3よりA2'の画像生成時間が短い。これは、ソーティングによる枝刈効果がなく、ソーティングすること自体がオーバヘッドとなったことを示している。

・写真4の例では、A3とB1の画像生成時間が同じである。画像面上でのプリミティブの重なりが少ないとB1が有利となることを示している。

・写真4、5の例では、B1の交差判定回数がA4やA3より少ないにもかかわらず、計算時間はB1が多い。これは、B1では、プリミティブのグルーピングに多くの時間を割かれるためである。逆に、提案アルゴリズムでは、部分木を動的に作成するコストが高くないことを示している。

・A2において、BVとプリミティブの交差判定回数をみるとわかるように、ソーティングを用いることにより、木の各ノードレベルでBV選択の一意性が近似的に成り立っていることがわかる。

(2) 小領域の大きさ

小領域の大きさと、画像生成時間の関係を調べる。表2に結果を示す。小領域が大きくなるとその中に入るプリミティブの数が大きくなり、部分木が大きくなって、交差判定回数が増える。一方小領域が小さくなると部分木は小さくなるが小領域の数が増えるので部分木を作るコストが大きくなる。したがって、小領域の大きさに最適点がある。表2の結果はそれを裏付ける。最適点はデータに依存するが、8×8の小領域が概ねよい結果を与えることが実験からわかる。

4. むすび

新しい光線追跡手法を提案し、その高速化効果を検証した。この手法は、与えられた木構造データから動的に部分木を作り、その木に対して、ソーティングと領域の一様性テストを併用して光線追跡を行う手法である。部分木を用いることにより、木の高さを減少させることができ、木の探索コストが低くなる。ソーティングは、木の枝刈効率を高める。一様性テストは領域内でレンダリングすべきプリミティブがただ1個であることをプリミティブが凸のときに保証する。部分木が高速化に寄与する効果の大きいことが実証された。また、これらの手法を組み合わせることにより極めて効率の良い光線追跡アルゴリズムを構成できることが示された。光線追跡法の計算コストは、交差判定コストの占める割合が極めて大きいことが指摘されているが、本手法はこの交差判定回数を著しく減少させる。

さらに、従来、高速と言われているArvoらの手法と比較しても本手法は高速であることを計算機実験により示した。

今後の課題として、2次光線に対する有効性の検証が残されている。また、本アルゴリズムの主要な計算は3次元ベクトル演算であることがプログラミングの結果わかっているので、その並列実行機構を内蔵した計算機SIGHT[4](当研究所で開発した計算機で、光線追跡法の高速実行をねらった計算機)に本アルゴリズムをインプリメントし、ハードとソフトの両面からの高速化効果を検証することが興味ある課題として残されている。

謝辞

日頃ご指導頂く知能ロボット研究部高野陸男部長はじめ研究部の各氏に感謝する。

参考文献

(1)Whitted T.:"An Improved Illumination Model for Shaded Display", Comm. ACM, Vol.23, No.6, pp.343-349, Jun. 1980
(2)Arvo J., Kirk D.:"Fast Ray Tracing by ray Classification", Comput. Graphics, Vol.21, No.4, pp.55-64, Jul. 1987
(3)Shinya M., Takahashi T., Naito S.:"Principles and Applications of Pencil Tracing", Comput. Graphics, Vol.21, No.4, pp.45-54, Jul. 1987
(4)Naruse T., Yoshida M., Takahashi T., Naito S.:"SIGHT - A Dedicated Computer Graphics Machine", Comput. Graphics Forum, Vol.6, No.4, pp.327-334, 1987