領域分割を用いた2次元光線交差問題の解法†

田口 東†† 飯島裕一†††

3次元物体の画像表示法として，反射，屈折まで考慮して非常にリアルな画像を得ることができる光線追跡法 (ray tracing method) がよく知られている．この手法の欠点は，光線と物体との交点の計算を数多く行わなければならないため，計算時間がかかることである．本論文では前処理によって空間をいくつかの領域に分割することにより，この計算を高速化することを考えた．対象としたのは，平面上に辺が平行となるように長方形が配置されている場合である．前処理による分割のねらいは，領域の境界線上に長方形を整列させることである．そして，光線と長方形との交点を二分探索によって求めることにより，近似的に光線の通る道筋にある物体だけを調べるようにした．計算機実験を行った結果，実験に用いたデータではほぼ長方形数の平方根に比例する計算時間で交点計算を行うことができ，この方法が有効であることが確かめられた．

1. はじめに

現在まで，光線追跡法[8]の高速化に関係して多くの研究がなされてきた．その主なものは，対象物を取り囲む簡単な形状の領域を考え，それと光線との交点の有無を先に調べ全体の交点計算の回数を減らす方法[6]，近接した点の類似性を利用する方法[7]，物体の形状が手続き的に定義されている場合にその手続きの性質を利用する方法[4]，対象物の階層構造[2]を利用する方法[7]，空間を分割する方法等である[3),6]．空間を分割する方法では，あらかじめ各部分領域内に含まれる物体を登録しておき，光線が通過する部分領域を順に選び，そこに登録された物体との交点計算を行う．これにより，光線の方向から離れている物体は無視され，高速化がはかれる．しかし，この方法で注意しなくてはいけないのは，一つの部分領域内での交点計算を減らすために分割数を細かくすると，領域をたどる手間が無視できなくなり，逆に分割が粗すぎると分割をした意味がなくなることである．

そこで，本論文では，各部分領域に含まれる物体はすべて境界線上に整列するような分割を考え，光線が境界線と交差する際に，交点計算の対象となる物体を二分探索で選び出せるようにした．また，その領域内では交点がない場合に，次に通過するはずの領域に関する情報も得られるようにした．対象としたのは，平面上に辺が平行となるように長方形が置かれている場合である．光線追跡法に応用するためには，あらかじめ空間を適当な平面に射影しておく必要があるが，これについては最後に述べることにする．

2. 光線追跡法の前処理としての空間の分割

図1に示すように x 軸と y 軸をとり，単位正方形内に長方形の辺が軸に平行となるように置かれているものとする．単位正方形の頂点を O_1, O_2, O_3, O_4 とする．空間の分割の仕方は，光線の向きを（x 正，y 正），（x 正，y 負），（x 負，y 正），（x 負，y 負）の四つの場合を考え，それぞれに対応して異なるデータを用意する．光線が軸に平行な場合には正負いずれと考えてもよい．以下では（x 正，y 負）の場合を説明する．

空間分割の基本的な考え方を説明するために，長方形は非常に小さく左上隅点で代表されるとする．最も左にある長方形（R_1 とする）を選び，R_1 から x 軸に垂線をおろし，その足を Q_1 とする．つぎに，R_1 より上にある長方形の中で最も左にあるもの（R_2 とする）を選び，R_2 からおろした垂線と，R_1 から右へ x 軸に平行に出した直線との交点 Q_2 を求める．R_2 に対しても同様に，R_2 より上で最も左にある長方形を R_3 とし，R_2 からの直線と R_3 からの垂線との交点 Q_3 を求める．順にこの操作を続け，最後に最も y 座標の大きい長方形 R_k が選ばれ，R_k から x 軸に平行に直線を出し，正方形の辺 O_2O_3 との交点 Q_{k+1} を求める．

線分列 $L=O_1Q_1$ Q_1R_1 R_1Q_2 Q_2R_2 … Q_kR_k R_kQ_{k+1} $Q_{k+1}O_3$ と $U=O_1O_4$ O_4O_3 とに囲まれる領域 B_1 には長方形は存在せず，L 上で長方形は x 座標（y 座標）の順に並んでいることに注意しよう．長方形

† Computational Method for 2-Dimensional Ray Intersection Problem Using Space Partition by AZUMA TAGUCHI (Department of Computer Science, Faculty of Engineering, Yamanashi University) and YUUICHI IIJIMA (NEC Corporation).

†† 山梨大学工学部計算機科学科
††† 日本電気(株)

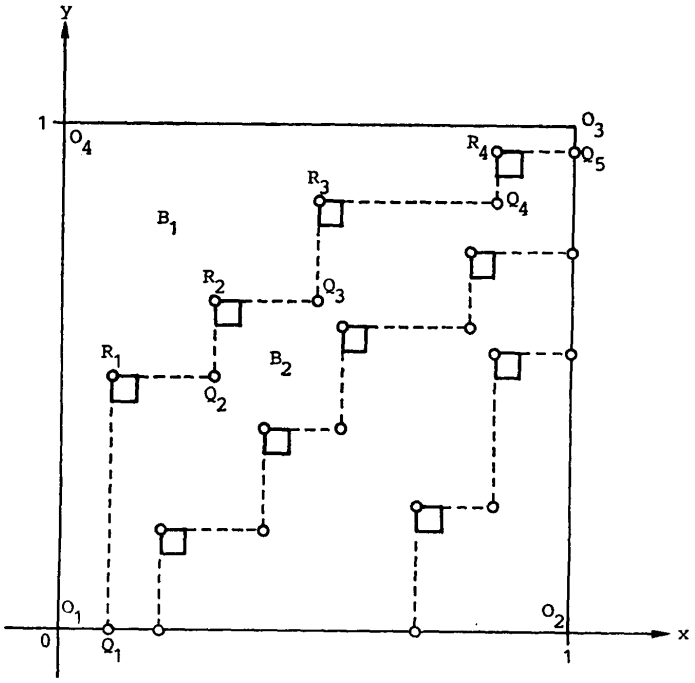

図 1　単位正方形内に置かれた"小さい"長方形
Fig. 1　Objects (small rectangles) in unit square and division of unit square.

$R_1, R_2, \cdots, R_k$ を取り除いた残りの長方形に対して同様の操作を行うことにより，B_1 と同じ性質を持つ領域 B_2 が得られる．これを繰り返して，単位正方形を長方形の頂点を境界線に利用した領域に分割することができる．

図1で B_1 内の点から右下に進む光線を考えよう．この分割を利用すると，たかだか3回の境界線（破線）と光線との交点計算と，その交点が長方形の辺上にあるかどうかの判定が必要である．境界線と光線との交点計算を後述のように速く行えるようにしておけば，何も工夫しない場合に10個の長方形すべてとの交点計算をするよりは効率がよいと考えられる．

有限な大きさの長方形に対しては，x 軸または y 軸に平行にのばした直線が他の長方形と交差したり，長方形が重なったりする場合も考慮しなければならない．簡単のために，以下に現れるすべての点の座標値は異なるものとする．

[領域の境界線上の頂点を作るアルゴリズム AL 1]

（入力）　単位正方形内に置かれた長方形 R_i $(i=1, \cdots, n)$．R_i の左上隅点を P_i，右上隅点を H_i，左下隅点を V_i とする．点Pの x 座標を $P(x)$，y 座標を $P(y)$ のように表す（図2参照）．

（出力）　点リスト $Lh(i)$, $Lv(i)$ $(i=1, \cdots, n)$ を作る．

$Lh(i)$：P_i から H_i へのばした直線上に並ぶ点のリスト．x 座標の上昇順にソートされる．

$Lv(i)$：P_i から V_i にのばした直線上に並ぶ点のリスト．y 座標の下降順にソートされる．

1°　P_i を x 座標の上昇順にソートしたリスト Lx，y 座標の上昇順にソートしたリスト Ly をつくる．

2°　$Lh(i)=Lv(i)=\{P_i\}$ $(i=1, \cdots, n)$ とする．すべての長方形の対 R_i, R_j $(i \neq j)$ に対して，辺 P_iH_i と辺 P_jV_j が交わるとき，その交点をリスト $Lh(i)$ と $Lv(j)$ に加える（図2 A_1, A_2）．

3°　長方形 R_i $(i=1, \cdots, n)$ に対して，P_i から H_i へ直線をのばしたとき，最初に交差する長方形の辺，もしなければ辺 O_2O_3，との交点を $\bar{Q}_i$，交差する長方形の番号を r_i とする．
$flag_i=off$ $(i=1, \cdots, n)$ とする．

4°　リスト Lx の先頭から順に次の操作を行う．
k 番目の要素 $P_a(x)$ を選んだとする．
P_a から V_a へ直線をのばしたとき，それと交点を持つ x 軸に平行な直線を出す長方形 R_b を
$P_b(y)=\max\{P_i(y) \mid flag_i=off,\ P_i(y)<V_a(y),$

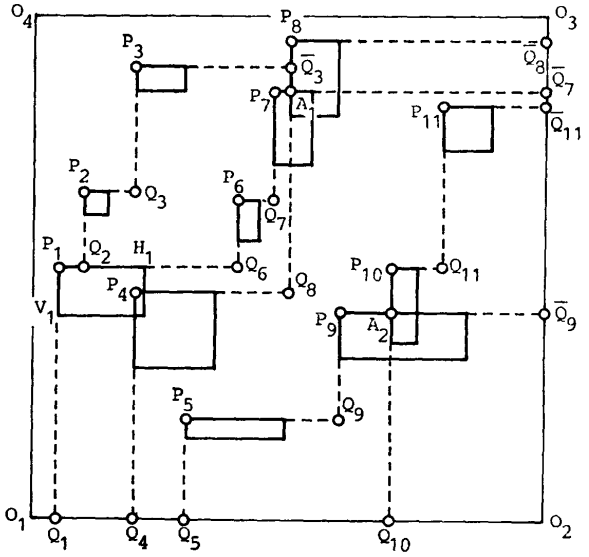

図 2　アルゴリズム AL 1 で生成された点と線分
Fig. 2　Points and line segments generated by AL 1.

$\bar{Q}_i(x)>P_a(x)>P_i(x)\}$

として選ぶ．もし，このような点 P_b がなければ，正方形の辺 O_1O_2 を長方形 R_b の代わりとする．$Q=(P_a(x), P_b(y))$ とすると，線分 P_bQ, QP_a は境界となるのでQを Lv(a), Lh(b) に加える．点Qが辺 P_bH_b 上になければ，$flag_b$=on とする．

5° リスト Ly の先頭から順に，長方形 R_i に対して，$flag_i$=off ならば点 $\bar{Q}_i$ を Lh(i) と Lv(r_i) に加える（図2の点 $\bar{Q}_3$, $\bar{Q}_7$, $\bar{Q}_8$, $\bar{Q}_9$, $\bar{Q}_{11}$）．

このアルゴリズムによって得られた点リスト Lh(i), Lv(i) (i=1, …, n) と，単位正方形の辺上の交点から，点間の線分を辺，線分の両端点を頂点とする平面グラフを作る．このグラフを基にして領域分割を行う．ここで，一つの領域の境界をなす辺上を x 座標または y 座標が増加する方向に進むとき，その領域を右にみる辺が作る境界を上の境界，左にみる辺が作る境界を下の境界と呼ぶことにする．

アルゴリズム AL 1 から作られる平面グラフの頂点における辺の交差の仕方は，図3の5個の型に分けられる．図中に一つの領域の境界を構成するための始点，終点，途中の点での辺の選び方の規則を示す．

［単位正方形を分割するアルゴリズム AL 2］

（入力） AL 1 から作られる平面グラフ．

（出力） 分割によってできる領域 B_k, k=1, …, m_b.

Eu(k)：領域 B_k の上の境界の辺のリスト．

El(k)：領域 B_k の下の境界の辺のリスト．

B_e：辺eを上の境界の一部とする領域．

R_e：辺eのすぐ右にある長方形．

1° スタック S={O_1} とする．k=1 とする．

2° スタックSの上端の点を取り出し出発点とする．スタックが空のとき終了する．

3° 上の境界を作るために，出発点から図3(a)の矢印に従って終点となる点まで進み，通過した辺のリスト Eu(k) を作る．通過した辺eに対して，領域と長方形を B_e, R_e に登録する．

4° 下の境界を作るために，3°と同じ出発点から図3(b)の矢印に従って終点まで進み，通過した辺のリスト El(k) を作る．途中の点で出発点となり得る点をスタックSに加える．2° へ戻る．

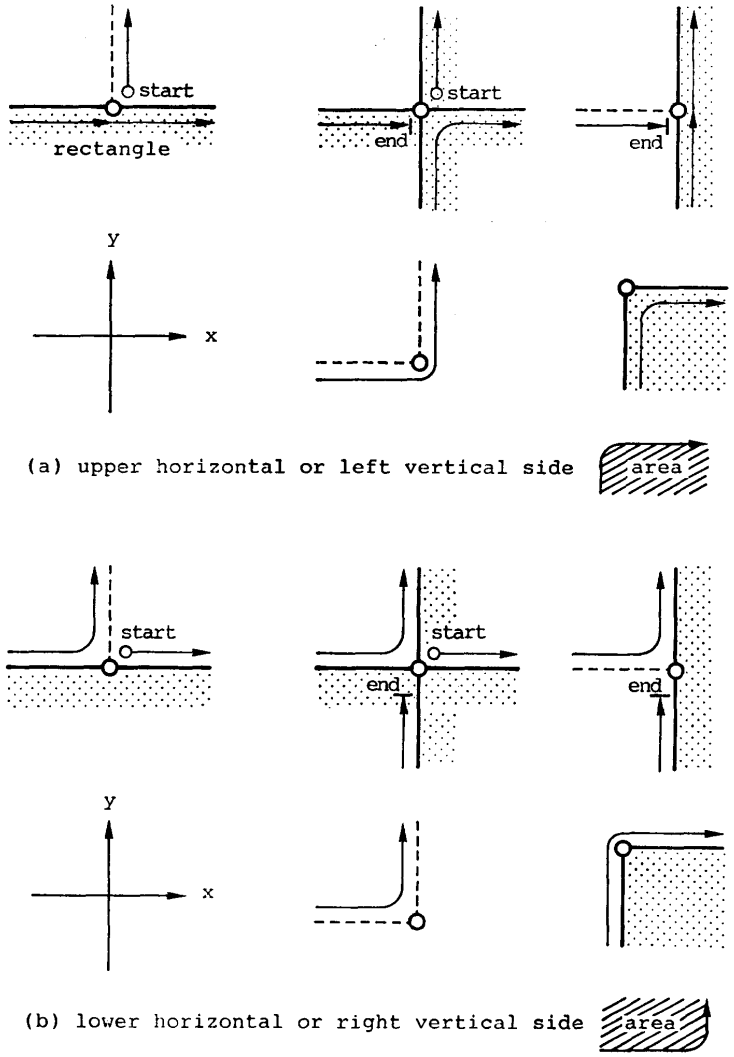

図3 頂点における辺の交差の型と境界を構成するための辺の選び方
Fig. 3 Types of intersections of edges at a point and rules for choosing adjacent edge to construct the boundary of an area.

図4に示すように，領域 B_k の上の境界に光線が到着したとき，次に交差する可能性のある辺は，その交点より x 座標が大きく y 座標が小さい範囲に限られ，このような辺はリスト El(k) 上で連続している．そこで，下の境界辺を延長して上の境界辺との交点を求め，それらの交点によって上の境界辺を区分する．そして，区分された線分に対して，次に交差する可能性のある下の境界辺の，リスト El(k) での最初と最後の位置を登録する．これにより，各辺eに対して区分さ

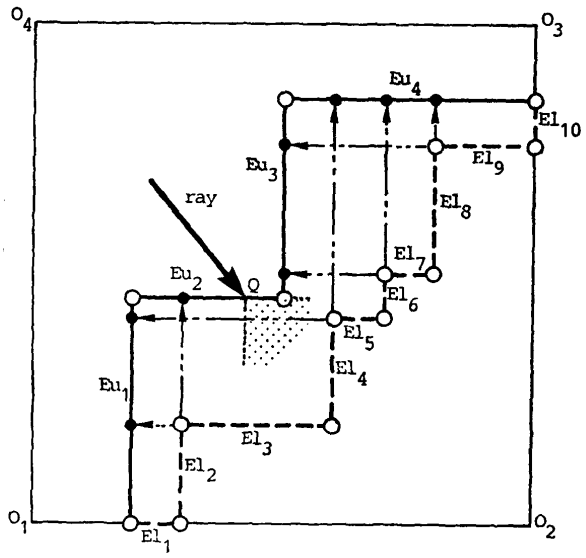

図 4 上の境界辺の区分

Fig. 4 Division of edges on upper horizontal or left vertical side. Ray intersecting edge Eu_3 at point Q will meet one of the edges El_3~El_6.

れた線分のリスト Mf(e), その区分に対応して El(k) での最初と最後の位置を与えるリスト Ms(e), Mt(e) を得る.

ここで, 前処理アルゴリズムの計算量と必要な記憶領域の大きさを見積っておこう. アルゴリズム AL1 において, 二つの長方形の辺が交差してできる交点数を n_1, 4° と 5° でできる交点のうちで長方形と単位正方形の辺上にない点の数を n_2 とすると, それらの辺上の点の数は $2(n-n_2)$ であるから, $P_i(i=1, \cdots, n)$ と単位正方形の頂点を加えて, 全頂点数は

$$m_p=3n+n_1-n_2+4$$

となる. 次に, 辺の数は, 長方形からのばした直線が $2n$ 本あり, 長方形と単位正方形の辺上の点一つによって1本, 長方形の辺の交点一つによって2本増えるので, 単位正方形の辺を加えて, 全辺数は

$$m_e=4n+2n_1-2n_2+4$$

となる. したがって, 領域の数は次のようになる.

$$m_b=n+n_1-n_2+1.$$

ここで, $n_2<n$ であるから, n_1 が小さい場合には, 頂点数と辺数はほぼ n に比例する. また, 領域数は長方形が非常に小さく, 一様に分布しているとみなせる場合には, 平均的に $\sqrt{2n}$ となる (付録).

アルゴリズム AL1 の計算時間は, $O(n^2)$ であり, AL2 の計算時間は $O(m_e)$ である.

以上の前処理のほかに, 光線の始点が含まれる領域を決定する点位置決定問題[5] を解く必要があり, そのための前処理が必要となる. これには Asano[1] の方法を用いた. この前処理の計算時間は平均的に $O(m_p)$ であり, 記憶領域も平均的に $O(m_p)$ である.

3. 光線追跡のアルゴリズム

2章で述べたデータ構造を用いて, 次のようなアルゴリズムを構成する.

[光線追跡のアルゴリズム TR1]

(入力) 2章によって作られるデータ構造.
光線の始点と x 方向, y 方向の増分.

(出力) 光線が交差する長方形の番号と交点.

1° 与えられた光線の x 方向, y 方向の増分に対応したデータ構造を選ぶ (以下 x 正, y 負とする).

2° 光線の始点を含む領域を見つける. B_k とする.

3° 領域 B_k の下の境界辺のリスト El(k) に対して二分探索を行い, 光線が交差する辺を見つける. その辺を e とし, 交点を Q とする.

4° 点Qが長方形 R_e の辺上にあるか, または辺 e が単位正方形の辺上にあれば終了する. そうでなければ, $B_k=B_e$ として, 次の領域に移る.

5° 辺 e の区分リスト Mf(e) に対して二分探索を行い, 点Qが属する区分を見つけ, Ms(e), Mt(e) から, 次に交差する可能性のある下の境界辺リストの最初の位置 s と最後の位置 t を得る.

6° 下の境界辺リスト El(k) の s と t の間を二分探索し, 光線と交差する辺 e を見つける. 4° へ行く.

このアルゴリズムの効率は, 点位置決定問題, 一つの領域を通過するのに必要な計算時間, 終了までに通過する領域の数に依存する. 点位置決定問題に用いた解法は平均的に $O(1)$ で問題を解く. 応用として考えている光線追跡法では, 光線の始点は視点または光線が交差した物体上の点であるから, すでにこの問題は解かれていると考えてもよい. 一つの領域を通過するための計算時間はたかだか $O(\log$ (境界上の辺数)) であり, 次に交差する辺の範囲を限定しているのでもっと少ない. 通過領域数の推定は難しいが, 光線の経路長を一定とすると, 全領域数に比例すると考えられるから, 長方形が小さく一様に分布している場合には $\sqrt{n}$ に比例する. このことは, 数値実験で確かめることにする.

4. 数 値 実 験

長方形の左上隅点の x 座標, y 座標を (0, 1) の一

様乱数を用いて定め，辺の長さを $(0,\ 2.5/n)$ の一様乱数を用いて定める．これは，n が十分大きいとき，辺の長さの平均値を h とすると，x 方向に dx 進む間に長方形と交差する確率が $hndx$ となることから，h を $1/n$ に比例するようにして，平均的に光線の経路長を n によらず一定とするためである．対象とする光線の進行方向は x 方向へ正，y 方向へ負に限定した．計算には山梨大学情報処理センター ACOS-850 を用いた．

比較のために，次の二つのアルゴリズムによっても計算した．TR2 はすべての長方形を総当たりに調べる最も素朴な方法，TR3 は長方形の頂点をソートしておき，光線の方向に応じて交点を持つ可能性のある範囲だけを取り出して，しかも一つの交点が求まった時に終了できるようにした方法である．

[光線追跡法 TR2]

(入力) 長方形 $R_i(i=1, \cdots, n)$．光線．

(出力) 光線が交差する長方形上の点 (x_q, y_q)．

1° 光線の始点を (x_1, y_1) とし，単位正方形から外へ出る点 (x_2, y_2) を求める．$(x_1<x_2,\ y_1>y_2)$ $x_q=x_2$, $y_q=y_2$ とする．

2° $i=1, \cdots, n$ に対して次の操作を行う．
長方形 R_i が $x_1<P_i(x)<x_q$ を満足し，$P_i(y)>y_2$ かつ $V_i(y)<y_1$ であれば，光線と直線 $x=P_i(x)$ との交点Qを計算する．点Qが辺 P_iV_i 上にあれば $x_q=Q(x)$, $y_q=Q(y)$ とする．

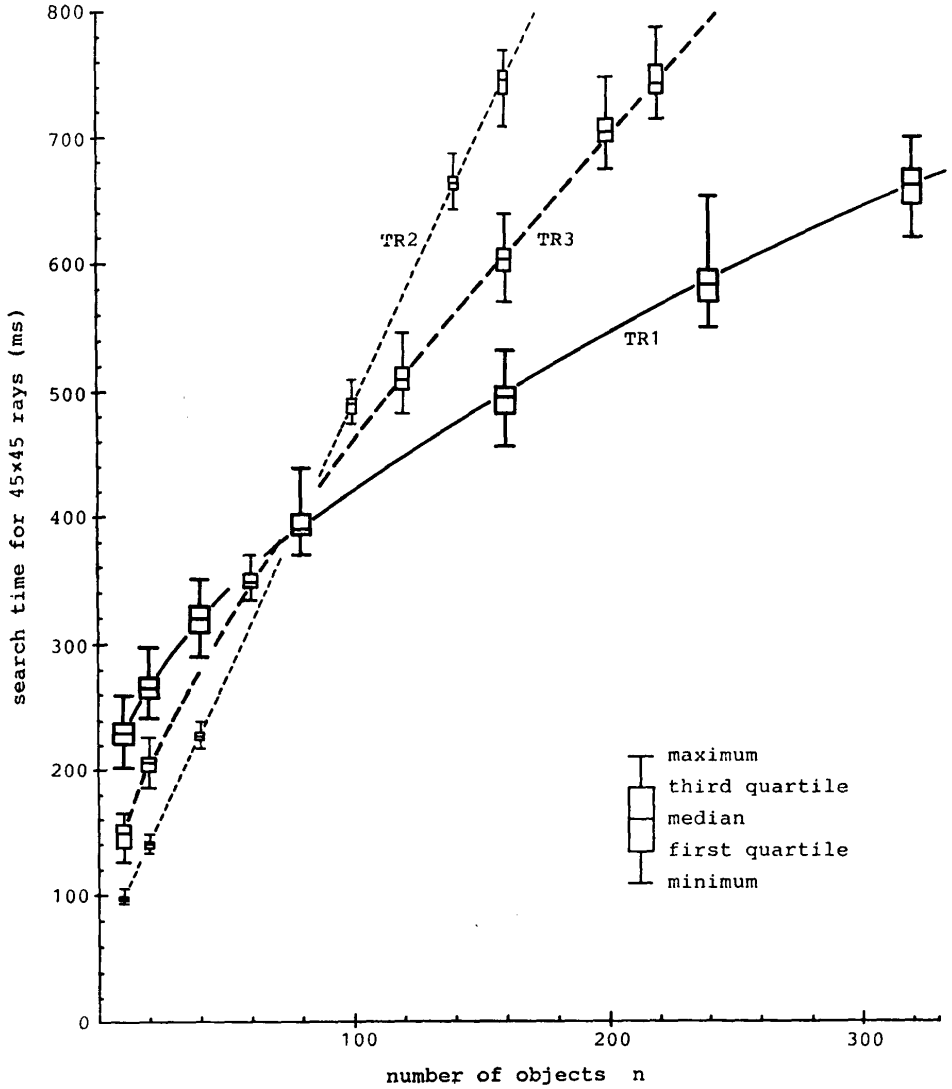

図 5 アルゴリズム TR 1, TR 2, TR 3 の交点計算時間の比較 (45×45 本の光線)
Fig. 5 Comparison of search times of algorithms TR 1, TR 2 and TR 3.

3° x 軸に平行な辺に対する処理に 2° を読み換える.

[光線追跡法 TR3]

(前処理)

1° 長方形の左上隅点 P_i の x 座標を上昇順にソートしたリスト Lx と, y 座標を下降順にソートしたリスト Ly をつくる.

(光線追跡)

1° TR2 の 1° と同じ.

2° リスト Lx の二分探索を行い $x_1<P(x)<x_q$ を満足する最小の要素 $P_s(x)$ を求める. リスト Lx 上で長方形 R_s から順に, 上の不等式を満足する範囲で長方形を選び, $P_i(y)>y_2$ かつ $V_i(y)<y_1$ であれば, 光線と直線 $x=P_i(x)$ との交点Qを計算する. 点Qが P_iV_i 上にあれば $x_q=Q(x)$, $y_q=Q(y)$ とし, 3° へ行く.

3° x 軸に平行な辺に対する処理に 2° を読み換える.

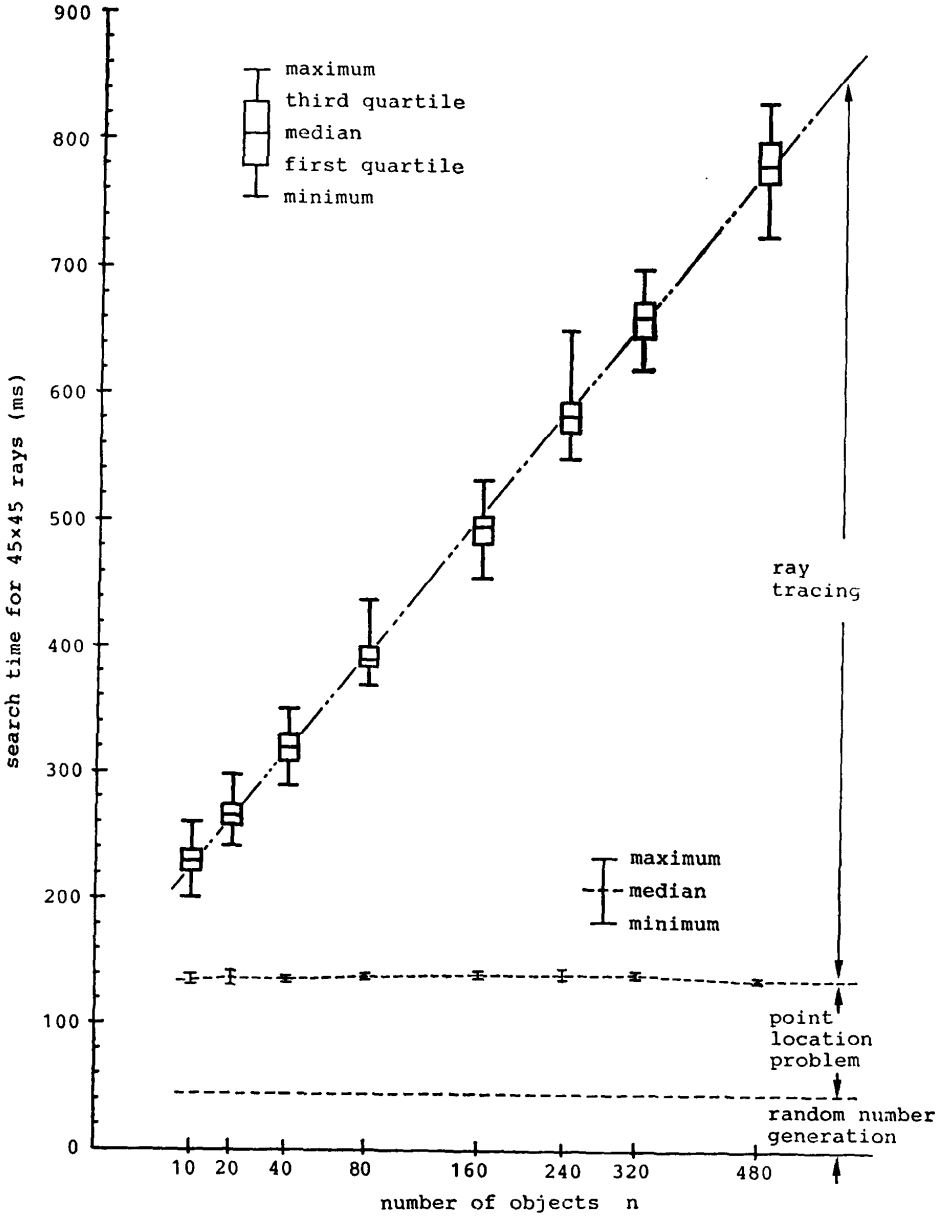

図 6 アルゴリズム TR1 の実行時間の内訳, 横軸の長さは $\sqrt{n}$ に比例

Fig. 6 Running times of ray tracing, point location and random number generation in TR1.

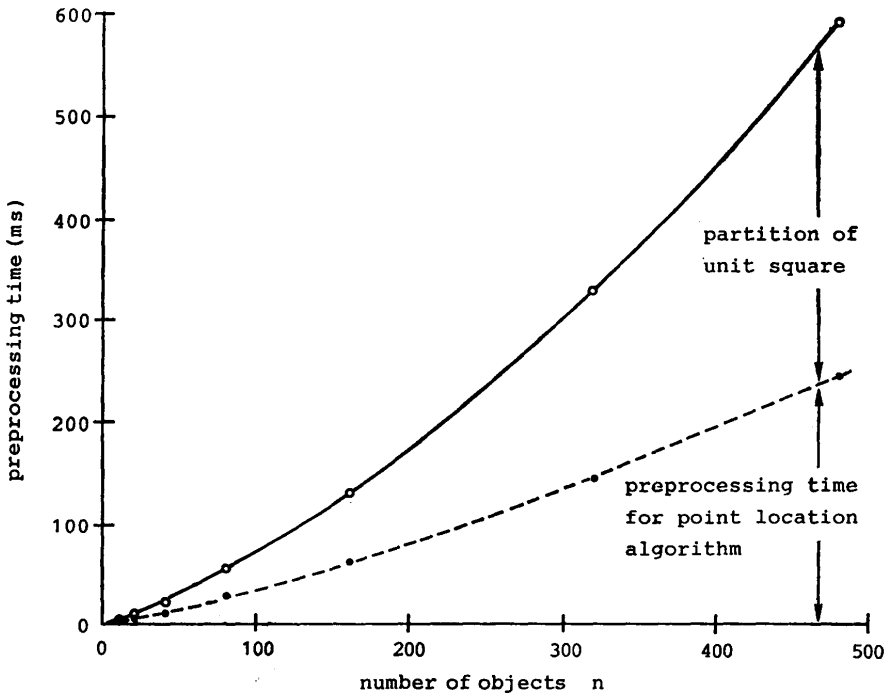

図 7 前処理時間
Fig. 7 Preprocessing time.

実験には，一つの n に対して乱数の初期値を変えて 50 組の長方形の配置のデータを作った．一つの配置について，光線の始点を均等間隔に 45×45 点とり，一つの始点から 1 本の光線を出し，その進行方向を x 方向は [0, 1)，y 方向は [−1, 0) に一様分布する乱数を用いて定めた．一つの長方形の配置に対して，45×45 本の光線すべての計算時間の合計を測定した．

図 5 に三つの光線追跡法による計算時間を示す．TR2 は長方形の数に比例して計算時間がかかっている．これは長方形を総当たりに調べているためである．TR3 では調べる長方形の数は TR2 よりも少ない．計算時間が n に比例していないのは，二分探索の $\log(n)$ の項が現れていると考えられる．本論文で提案する TR1 は $n=80$ くらいからまさっており，しかも計算時間の増加も緩やかである．

長方形数 n への依存を見るために，3 章で述べたことから，横軸を $\sqrt{n}$ として計算時間をプロットしたグラフを**図 6** に示す．この図には乱数発生，点位置決定問題に使われた計算時間も内訳として示してある．点位置決定問題の計算時間は n によらずほぼ一定であり，光線追跡に要した時間はほぼ $\sqrt{n}$ に比例していることがわかる．

前処理に要した時間を**図 7** に示す．単位正方形の分割に要した時間と，点位置決定問題の前処理に要した時間の内訳も示してある．

5. ま と め

平面上に長方形が配置されている場合に，前処理を行い光線追跡法を高速化するアルゴリズムを提案した．一つの長方形の配置に対して，光線の進行方向に応じた別々のデータを用意しなければならないという欠点があるが，交点計算回数は少なく，数値実験によっても有効な方法であることが確かめられた．

一般の形状の図形に対しては，それを取り囲むような長方形を考えれば適用可能である．また，3 次元空間に応用するためには，あらかじめ対象空間を適当な平面に射影しておき，光線を同じ平面上に射影して交点計算を行い，平面上で交点がある場合に，光線と物体の射影平面に垂直な方向の座標値を比較して交差判定を行うことになる．ただし，平面上で光線の始点を含んでしまうような物体があれば，それらを選び出し，別に交点計算を行わなければならない．

射影平面上での交差判定をできるだけ有効にするために，空間を平行な平面でいくつかの層に分け，それぞれの層で平面上への射影を行う方法，従来の領域分割法の部分領域内のデータ構造として利用する方法が

考えられる．これらについては，今後の課題として考察したいと考えている．

謝辞　日頃から厚い御指導をいただいている東京大学工学部計数工学科 伊理正夫教授に深く感謝いたします．

参考文献

1) Asano, T., Edahiro, M., Imai, H., Iri, M. and Muroto, K.: Practical Use of Bucketing Techniques in Computational Geometry, in Toussaint, G. T. (ed.), *Computational Geometry*, pp. 153-195, North-Holland, Amsterdam (1985).
2) Clark, J. H.: Hierarchical Geometric Models for Visible Surface Algorithms, *Comm. ACM*, Vol. 19, No. 10, pp. 547-554 (1976).
3) Glassner, A. S.: Space Subdivision for Fast Ray Tracing, *IEEE Comput. Gr. Appl.*, Vol. 4, No. 10, pp. 15-22 (1984).
4) Kajiya, J. T.: New Techniques for Ray Tracing Procedurally Defined Objects, *ACM Trans. Gr.*, Vol. 2, No. 3, pp. 161-181 (1983).
5) Lee, D. T. and Preparata, F. P.: Computational Geometry—A Survey, *IEEE Trans. Comput.*, Vol. c-33, No. 12, pp. 1072-1101 (1984).
6) Rubin, S. M. and Whitted, T.: A 3-Dimensional Representation of Fast Rendering of Complex Scenes, *SIGGRAPH '80 Proceedings*, pp. 110-116 (1980).
7) Weghorst, H., Hooper, G. and Greenberg, D. P.: Improved Computational Methods for Ray Tracing, *ACM Trans. Gr.*, Vol. 3, No. 1, pp. 52-69 (1984).
8) Whitted, T.: An Improved Illumination Model for Shaded Display, *Comm. ACM*, Vol. 23, No. 6, pp. 343-349 (1980).

付　録

領域の数の見積りを非常に簡単なモデルで行う．単位正方形内に n 個の点が一様に分布しているものとし，2章の最初に述べた空間の基本的な分割を考える．

x 座標の小さいほうから順に点を見ていって，領域を並行して作っていくものとする．点 $R(x_r, y_r)$ が現れたとき，$x=x_r$ に現れている領域の境界の線分（単位正方形の辺は除く）の中で，Rより下で最も近いものがあればその境界がRにつながる（図2 $R=R_2$ と線分 Q_2R_2）．もしRがすべての線分よりも下にあれば新しい領域ができる．領域数が k のとき，新しい点がすべての境界辺よりも下にある確率を $1/k$ とし，さらに，多少強引であるが，領域数が k となった後は，続く $k-1$ 個の点を処理した後に k 個目の点で領域が一つ増えるものとすると，領域数 m は次の不等式

$$\sum_{k=1}^{m-1} k < n < \sum_{k=1}^{m} k,$$

を満足し，m はほぼ $\sqrt{2n}$ となる．

（昭和61年7月16日受付）
（昭和61年10月8日採録）

田口　東（正会員）

昭和26年生．昭和49年東京大学工学部計数工学科卒業．同年三菱重工業(株)入社．昭和51年東京大学工学部助手，昭和55年山梨大学工学部講師を経て現在助教授（計算機科学科）．工学博士．ネットワーク理論，計算幾何学，数値計算などに興味を持つ．日本オペレーションズ・リサーチ学会，計測自動制御学会，日本音響学会，日本品質管理学会等各会員．

飯島　裕一（正会員）

昭和36年生．昭和59年山梨大学工学部計算機科学科卒業．昭和61年同大学院修士課程修了．同年日本電気(株)入社．情報処理基本ソフトウェア開発本部所属．