オクトツリーを利用した3次元物体の最近点探索アルゴリズム†

登尾啓史†† 畑 祐志††† 有本 卓††††

複数の物体が存在する3次元空間において，任意の点から最も近い点を物体上から選択することは多くの分野で有用であるが，これを実現するための効率的なアルゴリズムはこれまで提案されていない．そこで本論文では，そのような最近点を高速に選択するための実用可能なアルゴリズムを提案する．このアルゴリズムはオクトツリーというソリッドモデルのもとで機能する．物体の表面を含む領域のみを再帰的に8等分割することで作成されるオクトツリーは，物体の表面を位置に関して階層的に管理している．この階層構造を利用すると，1）任意の点から近い順に物体の表面を含む領域を選択し，さらに，2）最近点を獲得すれば直ちに終了するようなアルゴリズムが設計できる．このアルゴリズムは物体の表面を含む領域の大部分を探索しなくても最近点が決定できるので，その計算量は空間内の物体の個数やその形状の複雑さに依存しなくなる．したがって，このアルゴリズムは複雑な形状の物体が多数存在する空間においても高速に機能する．最後に，実験によって複雑な形状の空間におけるアルゴリズムの高速性，さらにはオクトツリーの最大レベルや最近点までの距離の変化に関するアルゴリズムの計算時間のふるまいを調べる．

1. はじめに

多数の点の中から任意の点に最も近いものを選択する問題は最近点問題 (closest-point problem) と呼ばれ，主に計算幾何学の分野で研究されている．計算幾何学においては，この問題をボロノイ図 (Voronoi diagram) を利用して解決しようとしており，2次元空間の点の集合に対するボロノイ図を作成する$O(n\log n)$, $O(n)$ の計算量のアルゴリズム[1)~4)]，また3次元空間の点の集合に対するボロノイ図を作成する $O(n^2)$ の計算量のアルゴリズムが提案されている[5),6)] (n：集合の点の個数)．これらのボロノイ図では，任意の点から最も近い点（以後最近点と記す）が高速に決定される．

3次元空間に存在する複数の物体の表面上から任意の点の最近点を選択することも一種の最近点問題であり，ロボティクスなどで必要な処理である．すなわち，ロボットの知能化の一つにコンピュータを利用したロボットの障害物回避経路の計画があり，そこでは障害物を膨らませることによってロボットを点で表現する技法が提案されている[7)]．点で表現されたロボットに対する膨らませた障害物上の最近点は，ロボットが衝突する可能性の最も高い障害物部分である．したがって，それを回避することで効率的なロボットの障害物回避アルゴリズムが作成できる．しかし，物体の集合に対してボロノイ図を作成するアルゴリズムはこれまでに提案されておらず，その作成も $O(n^2)$ (n：集合の面の個数) 程度の計算量では不可能であろうと予想されることもあって，そのような最近点を求める実用可能なアルゴリズムはいまだに提案されていない．

そこで本論文では，この最近点問題を解決するための2つの高速なアルゴリズムを提案しそれらを比較検討する．これらの最近点探索アルゴリズムはオクトツリーという最近点を求めるのに適したソリッドモデルのもとで機能する．オクトツリーは3次元空間のすべての物体を統一的に管理したソリッドモデルで，物体の表面が含まれる領域のみを再帰的に8等分割することで作成される[8)~11)]．このことから，オクトツリーは物体の表面上のすべての点を位置に関して階層的に管理する．

この階層構造を利用すると，1）任意の点から近い順に物体の表面を含む領域を選択し，2）最近点を得ると直ちに終了するアルゴリズムが設計できる．このアルゴリズムは物体の表面を含む領域の大多数を探索しなくても最近点が獲得できるので，その計算量は空間内の物体の個数やその形状の複雑さに依存しなくなる．それゆえに，自由形状を含む複雑な形状の物体が多数存在する空間においても，提案したアルゴリズムは高速に機能する．

---

† An Octree-Based Algorithm for a Closest-Point Problem of Three-Dimensional Solid Objects by HIROSHI NOBORIO (Department of Precision Engineering, Faculty of Engineering, Osaka Electro-Communication University), HIROSHI HATA (Toyota Motor Corporation) and SUGURU ARIMOTO (Department of Mathematical Engineering and Instrumentation Physics, Faculty of Engineering, University of Tokyo).

†† 大阪電気通信大学工学部精密工学科
††† トヨタ自動車(株)
†††† 東京大学工学部計数工学科

2. モデルの選択

3次元空間に存在する複数の物体の表面から任意の点Pの最近点P′を選択することは，物体の表面を位置に関してあらかじめ整理しておくことで効率的に行われる（**図1**）．そのため本研究では，複数の物体を含む空間そのものをオクトツリーというソリッドモデルで表現する．

物体の表面を含む領域のみを再帰的に8等分割することで作成されるオクトツリーは，物体の表面を領域（キューブ）の集合で表現するソリッドモデルであり，位置に関して階層的に物体の表面を管理している（**図2**）．3次元空間を表現するオクトツリーは以下のように作成される．まず，空間をオクトツリーのルートノードで表現する．次に，空間を8等分割したときに得られるサブ領域をルートノードの子ノードで表現する．このとき，サブ領域の位置と子ノードの番号は図2(b)のように対応づける．そして，サブ領域と物体の位置関係により子ノードにラベルをつける．すなわち，物体が存在しないサブ領域を白ノード，物体が占有するサブ領域を黒ノード，物体は存在するが占有しないサブ領域をミックスノードとする．ミックスノードに対応するすべての領域にこの処理を再帰的に行うと，3次元空間のオクトツリーが作成される．

ただし，ミックスノードのレベルがオクトツリーの最大レベル（物体を表現する近似精度）に達したとき，そのミックスノードを黒ノードにする．この処理によりすべての子ノードが黒ノードとなる親ノードが現れることがある．このときには，それらの子ノードを親ノードに統合したうえで親ノードを黒ノードに変更する．

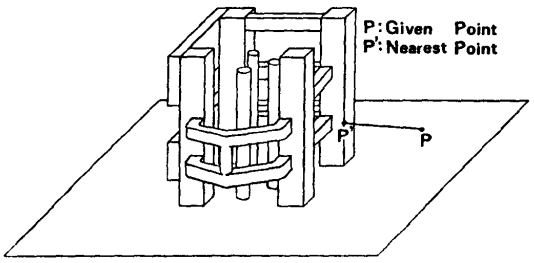


図1 3次元空間の最近点問題
Fig. 1 Closest-point problem of a three-dimensional space including several objects.

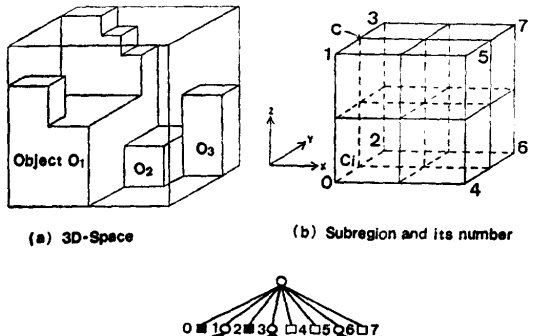


図2 3次元空間（a），サブ領域とその番号（b），空間を表現するオクトツリー（c）
Fig. 2 3D-space (a), a subregion and its number (b), and the octree of the 3D-space (c).

3. 修正 DF 探索にもとづく最近点探索法

ここでは，オクトツリーを縦型探索（DF 探索：Depth-First search）することで最近点を求めている[12]．一般に，オクトツリーの縦型探索はミックスノードの8つの子ノードをそれらの番号に従って再帰的に処理（これを**[基本処理]**と呼ぶ）することで達成されるが，ここでは最近点探索の高速化のため，8つの子ノードを点Pから近い順（これを**優先順位**と呼びこの決定法は後述する）に再帰的に処理するように修正する．この優先順位にもとづく DF 探索を修正 DF 探索と呼ぶことにする．

この方法では，距離 $D_m=\infty$ としてオクトツリーのルートノードに以下の[基本処理]を適用し，次々と現れる子ノードに優先順位にもとづいて[基本処理]を再帰的に適用していく．そして，ルートノードの[基本処理]の終了をもって最近点探索を終了させる．このとき，最近点は P_m，最短距離は D_m に保持される．**図3**のオクトツリーでは，斜線でハッチングされたノードがこの最近点探索法で探索される．

[基本処理]

この基本処理が適用されたノードの子ノードのうちまだ処理されていないものを優先順位の高い順（点Pに近い順）に選択して以下の処理を行う．すべての子ノードがすでに処理されているときには，適用されたノードの親ノードにおける[基本処理]にもどる．

1）子ノードが黒ノードの場合：子ノードに対応する領域上で点Pから最も近い点 P′ を選択する．これらの点Pと P′ の距離 D_p がすでに得られている最

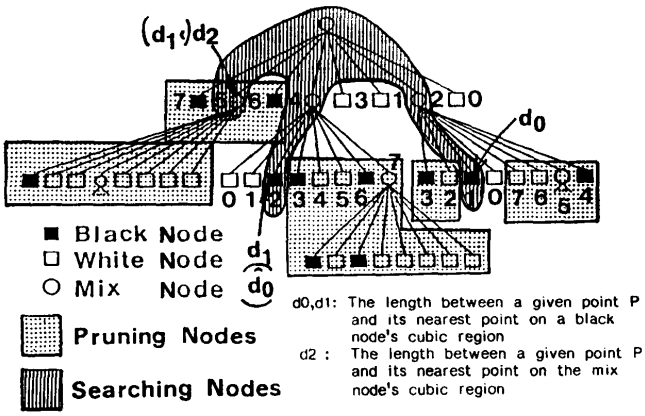


図 3　修正 DF 探索法によるオクトツリーの探索
Fig. 3　Search of the octree by the modified DF-search.

短距離 D_m よりも小さいときには，点 P′ は最近点になる可能性があるので，点 P′ を点 P_m，距離 D_p を最短距離 D_m として記録する．

この子ノードより優先順位が低い子ノードからは距離 D_p より小さな距離は得られないので，そのような子ノードは処理しなくてもよい．そこで，適用されたノードの親ノードの［基本処理］にもどる．このことから，図3のオクトツリーでは，ルートノードの子ノード2，4のもとの点でハッチングされたノードを処理しなくてもすむ．

2）子ノードがミックスノードの場合：子ノードに対応する領域上で点Pから最も近い点 P′ を選択する．これらの点Pと P′ の距離 D_p がすでに得られている最短距離 D_m よりも小さいときには，それに［基本処理］を適用する．ここで，この最近点探索法は再帰処理となる．

そうでないときには，この子ノードより優先順位が低い子ノードからは距離 D_p より小さい距離は得られないので，そのような子ノードは処理しなくてもよい．そこで，適用されたノードの親ノードの［基本処理］にもどる．このことから，図3のオクトツリーでは，ルートノードの子ノード5とその子孫ノード，さらには子ノード6，7などの点でハッチングされたノードを処理しなくてもすむ．

3）子ノードが白ノードの場合：現在の［基本処理］を続ける．

点Pの最近点 P′ はノードに対応する領域の面，稜，または頂点上に位置するが，これらは点Pとその領域の位置関係によって容易に判断され，いずれの場合においても点 P′ は高速に決定される（図4）．

〈優先順位の決定法〉

ノードに対応する領域の重心から点Pへのベクトルを (x, y, z)，その領域の一辺長さを $2D$ とする（図5）．

このとき，子ノードの領域と点Pの距離を各々計算しなくても，6つの子ノードの優先順位―点Pから近い順―はベクトルの成分の大きさ $|x|$, $|y|$, $|z|$ の大小を比較するだけで決定でき，さらに残りの2つの子ノードを含めた8つの子ノードの優先順位も，$|x|$, $|y|$, $|z|$ と D で記述された簡単な不等式を解くことにより決定される．これらは付録1で証明することにし，ここでは優先順位 Order[i] ($i=0,\cdots,7$) の決定法のみを説明する．

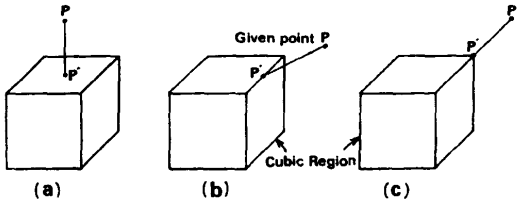
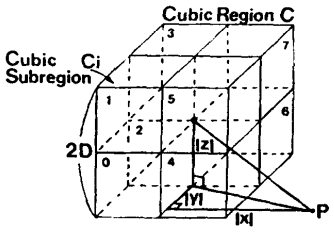


図 4　領域における最近点 P′ の位置，面上（a），稜線上（b），頂点上（c）
Fig. 4　The position of the closest-point P′ on a surface of a region (a), on an edge (b), on a vertex (c).



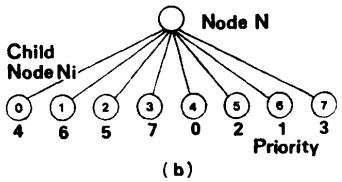


図 5　ノードNに対応する領域Cの重心と点Pの位置関係（a），サブ領域 C_i の優先順位（b）．
Fig. 5　The position relation between the center of gravity of the cubic region C and the point P (a), and the priority of the subregion C_i (b).

［ステップ1］ まず $S \leftarrow 0$ とし，$x \geqq 0$ ならば $S \leftarrow S+4$ とする．次に，$y \geqq 0$ ならば $S \leftarrow S+2$ とする．最後に，$z \geqq 0$ ならば $S \leftarrow S+1$ とする．このようにすると Order [0]$=S$ となる．

［ステップ2］ 関数 Max [i] ($i=1, 2, 3$) を $|x|$, $|y|$, $|z|$ のうち i 番目に大きな値と定義する．

また，関数 Number [i] を以下のように定義する．まず，Max [i]$=|x|, |y|, |z|$ のいずれかを選択する．次に，Max [i]$=|x|$ が選ばれたとき，$x \geqq 0$ ならば Number [i]$=4$，$x<0$ ならば Number [i]$=-4$ とする．Max [i]$=|y|$ が選ばれたとき，$y \geqq 0$ ならば Number [i]$=2$，$y<0$ ならば Number [i]$=-2$ とする．Max [i]$=|z|$ が選ばれたとき，$z \geqq 0$ ならば Number [i]$=1$，$z<0$ ならば Number [i]$=-1$ とする．

Order [1]$=$Order [0]$-$Number [3].

Order [2]$=$Order [0]$-$Number [2].

ここで，Max [3]$\geqq D$ のとき $h=2$ (Max [1]$-$Max [2]$-$Max [3])$+D$，Max [2]$\geqq D \geqq$Max [3] のとき $h=2D$ (Max [1]$-$Max [2])$-$Max [3]^2，Max [1]$\geqq D \geqq$Max [2] のとき $h=D$ (2Max [1]$-D$)$-$Max [2]$^2-$Max [3]^2，また，$D \geqq$Max [1] のとき $h=$Max [1]$^2-$Max [2]$^2-$Max [3]^2 とおく．

$h \geqq 0$ のとき，Order [3]$=7-$Order [0]$+$Number [1]，そうでないときには，Order [3]$=$Order [0]$-$Number [1] となる．

最後に，Order [i]$=7-$Order [$7-i$] ($i=4, \cdots, 7$) とする．

修正 DF 探索にもとづいたこの最近点探索法は，ミックスノードの8つの子ノードのみを優先順位にもとづいて探索するローカルな最適法である．それゆえ，修正 DF 探索法は最近点よりも遠い領域を探索したり近い領域を探索しないことがおこる．したがって，修正 DF 探索法の探索範囲は点 P から点 P′ までの最短距離 D を半径とした球とは必ずしも一致しない（図6）．

最後に，もし3次元空間が直方体でしか表現できないときにはノードが表す領域も直方体になる．このようなときでも，その直方体の X, Y, Z 方向のスケールを優先順位の決定法における h の計算式に導入することによりこの方法は同様に機能する．

4. BF 探索にもとづく最近点探索法

この方法は点Pからの距離が小さい順にまだ探索されていないオクトツリーのミックス・黒ノードを最良優先探索 (BF 探索：Best-First search) する[12]．まだ探索されていないノードは点Pからの距離が小さい順にオープンリストで管理される．

この方法では，まず，距離 $D_m=\infty$ としてオクトツリーのルートノードをオープンリストに登録し以下の［基本処理］を適用する．そして，次々に変化するオープンリストに［基本処理］を適用し，オープンリストの先頭ノードが黒ノードになれば最近点探索を終了させる．［基本処理］では，まずオープンリストの先頭ノードを選択する．次に，それがミックスノードであればその子ノードのうちのミックス・黒ノードを新たにオープンリストに登録し，黒ノードであればそれに対応する領域上から点Pに最も近い点 P′ を最近点 P_m として選択し，それらの距離を D_m に保持する．図7のオクトツリーでは，斜線でハッチングされたノードがこの方法で探索される．

［基本処理］

オープンリストの先頭ノードを取り出し以下の処理を行う．

1）先頭ノードが黒ノードの場合：先頭ノードに対応する領域上から点Pに最も近い点 P′ を最近点 P_m として選択し，それらの距離を最短距離 D_m として最近点探索を終了する．

2）先頭ノードがミックスノードの場合：先頭ノー

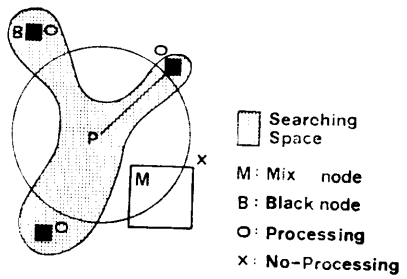


図 6 修正 DF 探索法の探索範囲
Fig. 6 Search region of the modified DF-search.

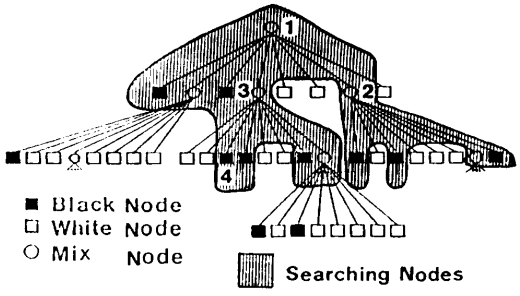


図 7 BF 探索法によるオクトツリーの探索
Fig. 7 Search of the octree by the BF-search.

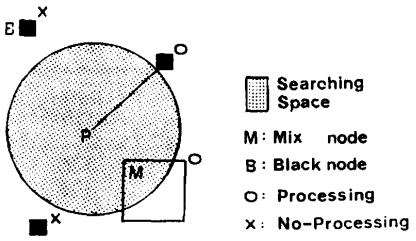


図 8 BF 探索法の探索範囲
Fig. 8 Search region of the BF-search.

ドの8つの子ノードからミックス・黒ノードを選択し，点Pからの距離に応じてそれらをオープンリストに登録する．オープンリストでは，すべてのノードは点Pからの距離の小さい順に並んでいるので，先頭からたどることで新たなノードは容易に登録できる．最後に，オープンリストが空でなければ再び［基本処理］を実行する．

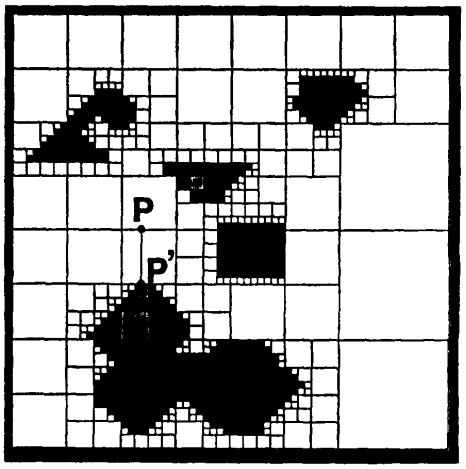
BF 探索にもとづいたこの最近点探索法は，点Pからの距離の小さい順にオクトツリーのノードを探索するグローバルな最適法である．それゆえ，BF 探索法は最近点より遠い領域は探索しないかわりにそうでない領域はすべて探索してしまう．したがって，BF 探索法の探索範囲は点Pから点 P′ までの最短距離Dを半径とした球と一致する（図8）．

5. 実験結果

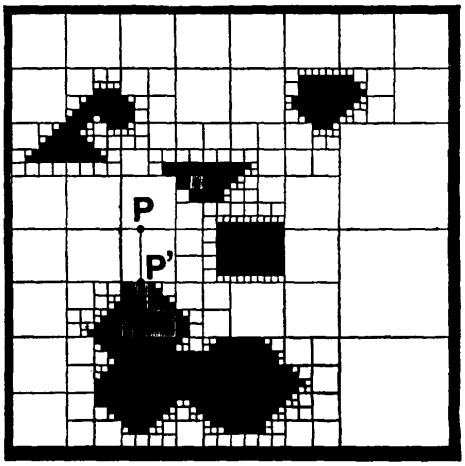
この章では，まず，2つの最近点探索法の計算量が3次元空間の物体の個数やその形状の複雑さに依存しないことを示す．次に，どちらの方法も複雑な形状の空間において高速に機能するが，修正 DF 探索法のほうが BF 探索法よりも効率的であることを示す．最後に，2つの最近点探索法の計算量がオクトツリーの最大レベルnや点Pと P′ の最短距離Dに依存することを示す．

5.1 空間の形状の複雑さに関する最近点探索法の計算量の評価

最近点探索法の探索範囲を明確にするため，ここでは2次元空間を表現したクワッドツリーを利用する．修正 DF 探索法が探索したクワッドツリーの黒ノードとレベル5のミックスノード，BF 探索法のオープンリスト内の黒ノードとレベル5のミックスノードを図9に示す．図9では，それらのうちの黒ノードの領域を点，ミックスノードの領域を斜線でハッチングしている．どちらの方法も点Pの周囲の物体の一部分しか探索していないので，それらの計算量は空間内の物体の個数やその形状の複雑さに依存しないことがわかる．



(a) DF-Search



(b) BF-Search

図 9 2次元空間（クワッドツリー）での最近点探索法の探索範囲
Fig. 9 Search region of each method in a two-dimensional space (quadtree).

5.2 2つの最近点探索法の計算時間の比較

複雑な形状の空間における実験で2つの最近点探索法の計算時間を比較する（図1）．この空間［0-1023, 0-1023, 0-1023］は最大レベル7のオクトツリーで表現されている（図 10）．点Pの最近点 P′ を求めるのに修正 DF・BF 探索法が要する計算時間を表1に示す．この結果は，どちらの最近点探索法も複雑な形状の空間において高速に機能すること，またオクトツリーをローカルに最適探索する修正 DF 探索法がオクトツリーをグローバルに最適探索する BF 探索法

表 1 複雑な形状の物体を含む3次元空間における最近点探索法の計算時間
Table 1 Calculation time of each method in the 3D-space including complicated shape objects.

Position			Distance	DF-Search (ms)	BF-Search (ms)	Ratio (DF/BF)
X	*Y*	*Z*				
624	488	360	0	7.2	18.9	0.38
624	488	366	6	7.2	18.8	0.38
510	530	80	6	23.9	30.5	0.78
500	500	2	34	24.4	56.7	0.43
510	230	93	90	11.1	27.2	0.41
500	500	500	124	46.7	98.9	0.47
845	673	12	136	36.1	69.4	0.52
425	161	283	159	18.9	45.0	0.42
305	531	580	186	53.9	100.0	0.54
391	832	532	202	25.5	47.8	0.53
981	500	500	296	53.9	75.5	0.71
548	20	355	300	21.1	46.1	0.46
212	916	520	302	16.1	27.8	0.58
660	981	218	309	40.5	83.3	0.49
830	43	508	334	15.5	28.3	0.55
204	14	560	372	15.5	28.9	0.54
20	51	38	429	16.7	37.8	0.44

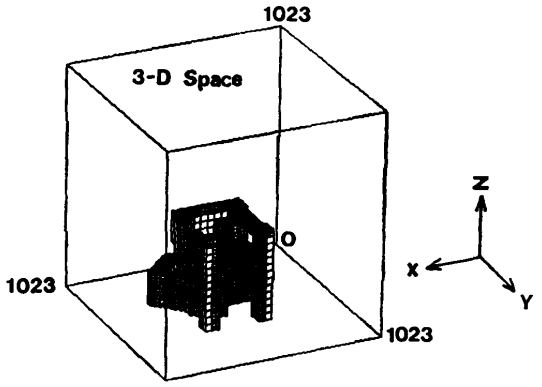


図 10 複雑な形状の物体を含む3次元空間のオクトツリー
Fig. 10 The octree of a three-dimensional space including complicated shape objects.

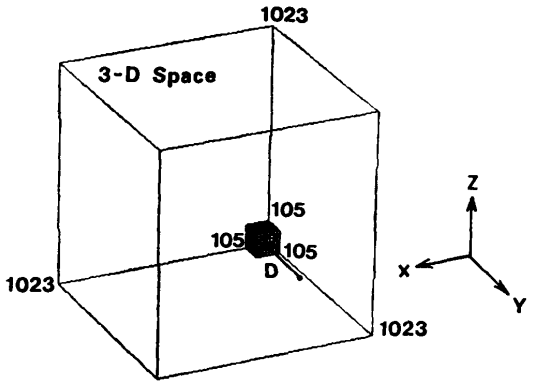


図 11 立方体の物体を含む3次元空間のオクトツリー
Fig. 11 The octree of a three-dimensional space including a cubic object.

よりも高速に機能することを示している.

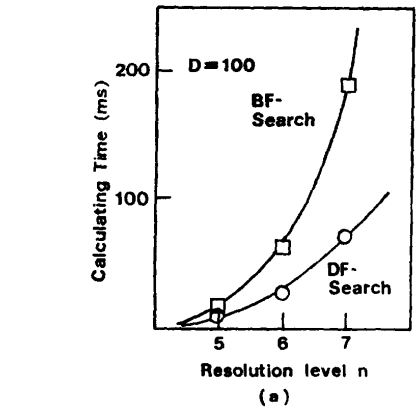
BF 探索法が修正 DF 探索法よりも計算時間がかかる理由としては, 1) 最近点より近いすべての領域を探索し, それらの領域と点Pの距離を必ず計算する; 2) その距離に応じてオープンリストのすべてのノードを逐次並びかえることがあげられる.

5.3 最大レベルや最短距離に対する最近点探索法の計算時間

オクトツリーの最大レベル n や点Pと P′ の最短距離 D の変化に対して最近点探索法の計算時間の変化を調べる. ここでは, 立方体 [0-105, 0-105, 0-105] の物体を含む空間 [0-1023, 0-1023, 0-1023] をオクトツリーで表現する (図 11).

オクトツリーの最大レベル n を増加させると, 物体の表面の領域がより細かく分割されそれらの個数は指数的に増加する. それゆえ, 最近点探索法が探索する物体の表面の領域, たとえば最近点周囲の領域の個数が指数的に増加することから最近点探索法の計算時間も指数的に増加する (図 12(a)).

また, 点Pの位置を移動させることで最短距離 D を増加させると, BF 探索法は動的に形成される距離 D を半径とする球Sと交差する領域, 特に最近点周囲の領域を探索する. ここで, 距離 D が増加すると球Sの曲率が徐々に大きくなり, 球Sと交差する最近点周囲の領域の個数も徐々に増加する. したがって, BF 探



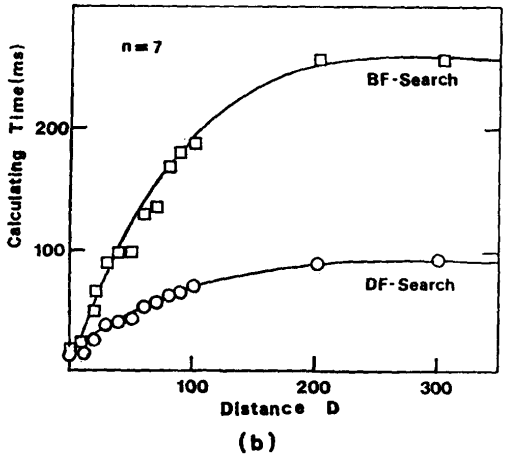


図 12 オクトツリーの最大レベル n (a), 最短距離 D (b) の変化に関する最近点探索法の計算時間
Fig. 12 Calculation time of each method as a function of the octree level n (a), and the minimum distance D (b).

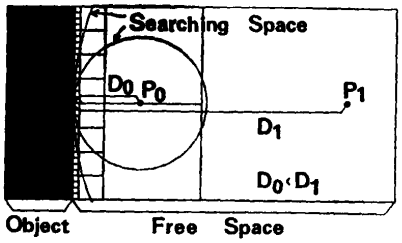


図 13 最近点探索法の探索範囲と距離 D の関係
Fig. 13 Search region of each method with respect to the minimum distance D.

索法の計算時間は徐々に増加する．修正 DF 探索法も同様の計算時間のふるまいをみせることから，最近点周囲で探索する領域の個数が徐々に増加しそれにつれて計算時間も増加するものと推測できる（図 12 (b)，図 13)．

どちらの最近点探索アルゴリズムもC言語で記述されており，すべての実験はワークステーション Apollo DN 590T の AEGIS 上で行っている．

6. 議　　論

ボロノイ図の作成が困難な3次元物体の最近点問題を解決するためには，最近点の決定に適したソリッドモデルを利用しなければならない．物体の表面を面の集合で定義する B-Reps というソリッドモデルで個々の物体を表現すると[13]，この最近点問題は点Pからすべての物体の面までの最短距離を各々計算し，そのなかで最小値をもたらす面の最近点 P′ を選択することで解決されるが，このアルゴリズムの計算量はすべての物体の面の個数に比例するので空間内の物体の個数やその形状の複雑さに直接依存する．したがって，複雑な形状の物体が多数存在する3次元空間においてそのアルゴリズムは利用できない．

このことから本論文では，点Pから近い順に物体の表面を探索することで物体の個数やその形状の複雑さに依存しない計算量をもつアルゴリズムを作成した．点Pを空間のどこにとってもそれから近い順に物体の表面を探索できるようにするためには，3次元空間においてその表面を位置に関してあらかじめ順序づけておく必要がある．このことを組織的に行うため，本研究では3次元空間をオクトツリーというソリッドモデルで表現している．物体の表面を含む領域のみを再帰的に8等分割することで作成されるオクトツリーは，すべての物体の表面を位置に関して階層的に管理したソリッドモデルであり，点Pから近い順に物体の表面が探索できる．

7. む　す　び

複雑な形状の物体が多数存在する3次元空間において任意の点Pの最近点 P′ を選択することは，多くの分野，たとえばロボティクスなどで必要であるにもかかわらず，実用可能なアルゴリズムはこれまで提案されていない．また，そのような空間を対象にしたボロノイ図の効率的な作成については手掛かりすらつかめていない．

そこで本論文では，そのような空間において最近点を高速に選択する実用可能なアルゴリズムを提案した．このアルゴリズムは最近点周囲の物体の表面しか探索しないので，複雑な形状の物体が多数存在する空間においても高速に機能する．また，オクトツリーは

自由形状を含むどのような形状の物体も柔軟に表現できるので，アルゴリズムはどのような形状の空間にも適用できる．

本論文では，修正 DF 探索ならびに BF 探索にもとづく2種類のアルゴリズムを提案した．前者のアルゴリズムはローカルな最適探索であり，後者はグローバルな最適探索である．しかし，BF 探索アルゴリズムは点Pとノードの領域との距離を逐次計算しなければならないうえ，その距離の小さい順にオープンリスト内のノードを繰り返し並び換えなければならないので計算時間がかかる．さらに，BF 探索アルゴリズムは膨大な個数のノードをオープンリストに保持しなければならないが，修正 DF 探索アルゴリズムは最大 $8n$ 個（n：オクトツリーの最大レベル）のノードを保持するだけでよい．これらのことから，修正 DF 探索アルゴリズムが BF 探索アルゴリズムよりも優れていることがわかる．

謝辞 実験に必要なワークステーションを利用させていただいた，(株)テイジイ総合研究所に感謝する．なお，本研究の一部は文部省科学研究費による．

参 考 文 献

1) Shamos, M. I. and Hoey, D.: Closest-Point Problems, *Proc. of the 16th IEEE Symposium Foundations of Computer Science*, pp. 151-162 (1975).
2) Green, P. J. and Sibson, R.: Computing Dirichlet Tessellation in the Plane, *Comput. J.*, Vol. 21, No. 2, pp. 168-173 (1978).
3) Ohya, T., Iri, M. and Murota, K.: A Fast Voronoi-Diagram Algorithm with Quaternary Tree Bucketing, *Inf. Process. Lett.*, Vol. 18, No. 4, pp. 227-231 (1984).
4) Ohya, T., Iri, M. and Murota, K.: Improvements of the Incremental Method for the Voronoi Diagram with Computational Comparison of Various Algorithms, *Journal of the Operations Research Society of Japan*, Vol. 27, pp. 306-337 (1984).
5) Tanemura, T. et al.: Geometric Analysis of Crystallization of the Softcore Model, *Progress of Theoretical Physics*, Vol. 58, No. 4, pp. 1079-1095 (1977).
6) Yamamoto, R. and Doyama, M.: The Polyhedron and Cavity Analysis of a Structural Model of Amorphous Iron, *J. Physics, Series F, Metal Physics*, Vol. 9, No. 4, pp. 617-627 (1979).
7) Udupa, S.: Collision Detection and Avoidance in Computer Controlled Manipulator, *Proc. of the 5th IJCAI*, pp. 737-748 (1977).
8) Jackins, C. L. and Tanimoto, S. L.: Oct-Trees and Their Use in Representing Three-Dimensional Objects, *Computer Vision, Graphics, and Image Processing*, Vol. 14, No. 3, pp. 249-270 (1980).
9) Meagher, D.: Geometric Modeling Using Octree Encoding, *Computer Vision, Graphics, and Image Processing*, Vol. 19, No. 2, pp. 129-147 (1982).
10) 登尾，福田，有本：BRep からオクトツリーへの変換アルゴリズムとその評価，情報処理学会論文誌，Vol. 28, No. 10, pp. 1003-1012 (1987).
11) 登尾，福田，有本：複数枚の画像を用いて3次元物体を近似したオクトツリーを生成する一手法，情報処理学会論文誌，Vol. 29, No. 2, pp. 178-189 (1988).
12) Nilsson, N. J.: *Principles of Artificial Intelligence*, Tioga, Palo Alto, Calif. (1980).
13) Requicha, A. A. G. and Voelcker, H. B.: Solid Modeling: Current Status and Research Directions, *IEEE Computer Graphics and Application*, Vol. 3, No. 7, pp. 25-37 (1983).

付録 1 8つの子ノードの優先順位の決定

8つの子ノードの優先順位の決定の正当性を証明する．$x \geqq y \geqq z \geqq 0$ と仮定しても，対称性より議論の一般性は失われない．図2のように，子ノードには0～7の番号がつけられており，任意の点Pから子ノード i に対応する領域までの距離を d_i と表現する．

(a) $|z| \geqq D$:

$$d_0^2 = |x|^2 + |y|^2 + |z|^2,$$
$$d_1^2 = |x|^2 + |y|^2 + (|z| - D)^2,$$
$$d_2^2 = |x|^2 + (|y| - D)^2 + |z|^2,$$
$$d_3^2 = |x|^2 + (|y| - D)^2 + (|z| - D)^2,$$
$$d_4^2 = (|x| - D)^2 + |y|^2 + |z|^2,$$
$$d_5^2 = (|x| - D)^2 + |y|^2 + (|z| - D)^2,$$
$$d_6^2 = (|x| - D)^2 + (|y| - D)^2 + |z|^2,$$
$$d_7^2 = (|x| - D)^2 + (|y| - D)^2 + (|z| - D)^2.$$

これら8つの式から，不等式 $d_0 \geqq d_1 \geqq d_2 \geqq d_3$, $d_4 \geqq d_5 \geqq d_6 \geqq d_7$, $d_2 \geqq d_4$, $d_3 \geqq d_5$ が得られ，Order [0]=7, Order [1]=6, Order [2]=5, Order [5]=2, Order [6]=1, Order [7]=0 となる．

ここで，$d_3^2 - d_4^2 = D\{2(|x| - |y| - |z|) + D\}$ より，$h = 2(|x| - |y| - |z|) + D \geqq 0$ のとき，$d_3 \geqq d_4$, つまり Order [3]=4, Order [4]=3 となる．そうでなければ，$d_4 \geqq d_3$, つまり Order [3]=3, Order [4]=4 となる．

(b) $|y|\geqq D\geqq|z|$:

$d_0^2=|x|^2+|y|^2+|z|^2$,
$d_1^2=|x|^2+|y|^2$
$\geqq|x|^2+(|y|-D)^2+D^2$,
$d_2^2=|x|^2+(|y|-D)^2+|z|^2$,
$d_3^2=|x|^2+(|y|-D)^2$,
$d_4^2=(|x|-D)^2+|y|^2+|z|^2$,
$d_5^2=(|x|-D)^2+|y|^2$
$\geqq(|x|-D)^2+(|y|-D)^2+D^2$,
$d_6^2=(|x|-D)^2+(|y|-D)^2+|z|^2$,
$d_7^2=(|x|-D)^2+(|y|-D)^2$.

これら8つの式から，不等式 $d_0\geqq d_1\geqq d_2\geqq d_3$, $d_4\geqq d_5\geqq d_6\geqq d_7$, $d_2\geqq d_4$, $d_3\geqq d_5$ が得られ，Order [0]=7, Order [1]=6, Order [2]=5, Order [5]=2, Order [6]=1, Order [7]=0 となり，$h=2D(|x|-|y|)-|z|^2\geqq 0$ のとき，$d_3\geqq d_4$，つまり Order [3]=4, Order [4]=3，そうでなければ，$d_4\geqq d_3$，つまり Order [3]=3, Order [4]=4 となる．

(c) $|x|\geqq D\geqq|y|$:

$d_0^2=|x|^2+|y|^2+|z|^2$,
$d_1^2=|x|^2+|y|^2$,
$d_2^2=|x|^2+|z|^2\geqq(|x|-D)^2+D^2+|z|^2$,
$d_3^2=|x|^2\geqq(|x|-D)^2+D^2$,
$d_4^2=(|x|-D)^2+|y|^2+|z|^2$,
$d_5^2=(|x|-D)^2+|y|^2$,
$d_6^2=(|x|-D)^2+|z|^2$,
$d_7^2=(|x|-D)^2$.

これら8つの式から，不等式 $d_0\geqq d_1\geqq d_2\geqq d_3$, $d_4\geqq d_5\geqq d_6\geqq d_7$, $d_2\geqq d_4$, $d_3\geqq d_5$ が得られ，Order [0]=7, Order [1]=6, Order [2]=5, Order [5]=2, Order [6]=1, Order [7]=0 となり，$h=D(2|x|-D)-|y|^2-|z|^2\geqq 0$ のとき，$d_3\geqq d_4$，つまり Order [3]=4, Order [4]=3，そうでなければ，$d_4\geqq d_3$，つまり Order [3]=3, Order [4]=4 となる．

(d) $D\geqq|x|$:

$d_0^2=|x|^2+|y|^2+|z|^2$,
$d_1^2=|x|^2+|y|^2$,
$d_2^2=|x|^2+|z|^2$,
$d_3^2=|x|^2$,
$d_4^2=|y|^2+|z|^2$,
$d_5^2=|y|^2$,
$d_6^2=|z|^2$,
$d_7^2=0$.

これら8つの式から，不等式 $d_0\geqq d_1\geqq d_2\geqq d_3$, $d_4\geqq d_5\geqq d_6\geqq d_7$, $d_2\geqq d_4$, $d_3\geqq d_5$ が得られ，Order [0]=7, Order [1]=6, Order [2]=5, Order [5]=2, Order [6]=1, Order [7]=0 となり，$h=|x|^2-|y|^2-|z|^2\geqq 0$ のとき，$d_3\geqq d_4$，つまり Order [3]=4, Order [4]=3，そうでなければ，$d_4\geqq d_3$，つまり Order [3]=3, Order [4]=4 となる．

最後に，$|x|=$Max [1], $|y|=$Max [2], $|z|=$Max [3] とおくと，8つの子ノードの優先順位の正当性が証明できる．

付録 2　提案したアルゴリズムで得られた最短距離と真の最短距離の差の評価

オクトツリーではすべての物体は黒ノードの集合で近似的に表現されている．それゆえ，点Pと黒ノード上の最近点 P′ がなす最短距離 d' は，点Pと物体上の最近点 P* がなす最短距離 d^* と異なる．ここでは，これらの距離 d' と d^* の差を評価する．アルゴリズムは黒ノードに対応する領域A上に最近点 $P_a'(=P')$ を選択して終了するとし，点P と P_a' の距離を d_a' $(=d')$ とする．

(1) 領域Aが点 P* を含む場合

点Pから領域A内の物体上の最近点 $P_a^*(=P^*)$ までの距離を $d_a^*(=d^*)$ と表す．距離 d_a' は d_a^* より小さく $(d_a'\leqq d_a^*)$，点 P_a' と P_a^* はどちらも領域A内に存在するので，距離 d_a' と d_a^* の差は領域Aの対角線の長さ l でおさえられる $(d_a^*-d_a'\leqq l)$，これらのことから，不等式 $0\leqq d^*-d'\leqq l$ が得られる．

(2) 領域Aが点 P* を含まない場合

まず，点 P* が存在する領域をBと表現する．点Pから領域B上の最近点 P_b' までの距離を d_b'，点Pから領域B内の物体上の最近点 P_b^* $(=P^*)$ までの距離を d_b^* $(=d^*)$ と表す（図 14）．ここで，点 P′ と P* はそれぞれ点 P_a' と P_b^* に対応し，不等式 $(d'$

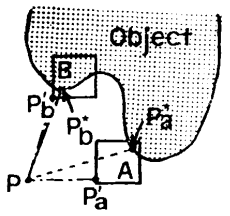


図 14　オクトツリー上の最近点 P′ と物体上の最近点 P* の関係

Fig. 14　Relation of the closest-point P′ on the octree and the closest-point P* on the objects.

$=)\ d_a' \leqq d_b'$, $d_b' \leqq d_b^*$, そして $(d^*=)\ d_b^* \leqq d_a^*$ が得られる．これらの不等式を組み合わせると，不等式 $d_a' = d' \leqq d^* \leqq d_a^*$ が得られ，この結果，$0 \leqq d^* - d' \leqq l$ となる．

一般に，領域Aのレベルはオクトツリーの最大レベル n であるので，その対角線の長さ l は $\sqrt{3}\,L/2^n$ となる（L：3次元空間の一辺長さ）．

（昭和63年6月17日受付）
（平成元年1月17日採録）

登尾　啓史（正会員）

昭和33年11月4日生．昭和57年静岡大学工学部情報工学科卒業．昭和59年同大学院工学研究科修士課程修了．昭和62年大阪大学大学院基礎工学研究科博士課程修了．大阪大学工学博士．同年同学機械工学科助手．昭和63年大阪電気通信大学工学部精密工学科講師．ロボティクス，コンピュータ・ビジョン，コンピュータ・グラフィックスにおける3次元モデルの研究に従事．電子情報通信学会，日本ロボット学会，IEEE 各会員．

畑　祐志

昭和39年7月29日生．昭和62年大阪大学基礎工学部機械工学科卒業．同年トヨタ自動車(株)入社．動力伝達系統の開発に従事．現在，駆動技術部計画課勤務．

有本　卓（正会員）

昭和11年8月3日生．昭和34年京都大学理学部卒業．沖電気工業(株)，東京大学工学部助手，講師を経て昭和43年大阪大学助教授，基礎工学部機械工学科に勤務，昭和48年同教授．昭和63年東京大学工学部計数工学科教授を併任．工学博士．ロボティクス，ディジタル信号処理，制御理論等の研究教育に従事．日本機械学会，計測自動制御学会，電子情報通信学会等の各会員．IEEE の Fellow 会員．