アルゴリズム 71−11
（2000. 1. 17）

GBD木における線分の分割法の改良

伊野 敦士† 増田 澄男‡ 山口 一章‡
† 神戸大学大学院自然科学研究科 ‡ 神戸大学工学部

あらまし 平面上の線分の集合を管理し得るデータ構造の一つとしてGBD木が知られている．GBD木により，線分の効率的な挿入・削除と範囲検索が可能であるが，長い線分が存在する場合に検索効率が低下することがある．この問題点に対処するため，長い線分を複数個に分割することが提案されているが，その方法は非常に単純なものであり，改善の余地が残されている．本研究では，線分の分割法にいくつかの簡単な工夫をすることによって，GBD木の検索効率をさらに改善できることを計算機実験により示す．

Improved Methods for Splitting Line Segments in a GBD Tree

Atsushi Ino†, Sumio Masuda‡, and Kazuaki Yamaguchi‡

† Graduate School of Science and Technology, Kobe University
‡ Faculty of Engineering, Kobe University

Abstract A GBD tree is a data structure that can maintain a set of line segments on a plane. By using it, we can efficiently perform three kinds of operations; insertion, deletion and range search. However, the search performance is not good enough if long line segments exist. In order to improve it, a method for splitting long line segments was proposed. In this article, we show that some modifications of the splitting method further improve the search performance.

1. まえがき

地理情報やCADデータなどの大規模な図形データを効率的に管理・検索するためのデータ構造には，対象平面（全データが存在する領域）を階層的に領域分割し，その分割過程を多分木で表すR木[1]，RMD木[2]，GBD木[3]，GBD木の改良版[4]（以下，改良GBD木と呼ぶ）などがある．

本研究では，様々な種類の図形データのうち，基本的で，かつ実用上頻繁に扱われる線分データについて考える．GBD木で線分データを管理する場合，各節点は対象平面のある部分領域に対応し，各内部節点（葉でない節点）は子のそれぞれについて領域式，外接長方形という2種類の情報を保持している．ここで領域式とは，領域の位置と大きさを表すビット表現であり，外接長方形とは，子孫の葉に記憶されている線分データのすべてを含み，かつ各辺が座標軸に平行であるような最小の長方形のことである．GBD木では，挿入・削除時に領域式を，範囲検索時に外接長方形をそれぞれ参照することにより，各々の処理の高速化を図っている．しかしながら，長い線分が存在する場合には，一般に，兄弟節点の外接長方形の重なりが多くなるため，検索効率が低下する．そのため改良GBD木では，長い線分のそれぞれをある一定の大きさ以下の複数の部分に分割した後に，GBD木に加えている．これにより，兄弟節点の外接長方形の重なりが一般に少なくなり，検索効率が向上する．しかしながら，この分割方法は非常に単純なものであり，改善の余地が残されている．また，長い線分が多く存在する場合に記憶領域が大幅に増加するという問題点ももつ．

本研究は，検索効率のさらなる向上を主な目

的として，線分のいくつかの分割法を新たに示し，計算機実験によりそれらの効果を調べるものである．

以下，2章でGBD木と改良GBD木の概略について説明する．3章では，改良GBD木の問題点と新しい分割法について述べる．4章では，提案した分割法と従来法を計算機実験により比較する．最後に，5章で本論文のまとめと今後の課題について述べる．

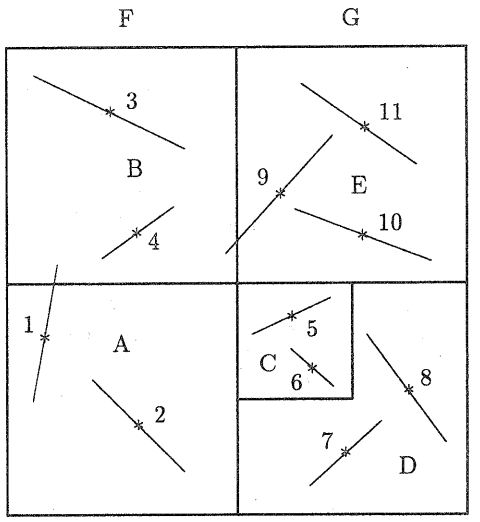
2. GBD木と改良GBD木の概略

2.1 GBD木の概略

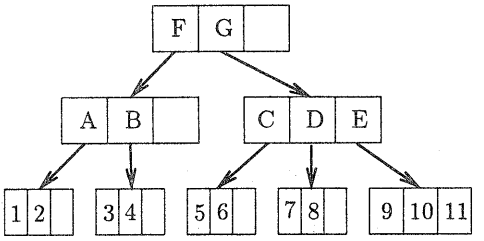
GBD木はBD木[5]をもとにしたデータ構造であり，対象平面を階層的に領域分割し，その分割過程を表した木構造により図形データを管理する．本研究では，図形データとして線分のみを考え，対象平面は正方形領域であるものとする．線分データに対するGBD木の例を図1に示す．同図(a)は領域分割の様子を，同図(b)は対応するGBD木を示している．図1(a)において，'＊'は各線分データの中心点を示している．また，全領域は太い実線で示すように5つの領域A～Eに分割されている．FはAとBを併せた領域であり，GはC～Eを併せた領域である．

GBD木の各節点は，対象平面のある部分領域に対応している．例えば図1(b)において，根は対象平面全体に，その左の子は領域Fに，左端の葉は領域Aに，それぞれ対応している．これ以降，記述を簡単にするために，領域とそれに対応する節点を同じ記号で表すことがある．

GBD木の各節点はある定数個（図1(b)では3）のスロットに分かれている．内部節点の各スロットは，子へのポインタと，その子の領域の領域式，および子の領域内に中心点をもつようなすべての線分を含む外接長方形の情報をもっている．また，葉の各スロットは，その葉の領域内に中心点をもつ1本の線分データへのポインタと，その線分の外接長方形の情報を保持している．ここで領域式と外接長方形について説明する．



(a) 領域分割



(b) GBD木

図1: GBD木の例

領域式： 対象平面に対して，分割軸を垂直→水平→垂直→…と交互に替えながら，面積2等分割を繰り返して得られる正方形，または縦横比が2：1の長方形の領域の位置と大きさを示すものであり，「0」,「1」および終了を示す記号「＊」で表される．「0」は領域を2等分割したときの座標原点に近い側の領域を，「1」は遠い側の領域をそれぞれ示している．例えば，図2に示した領域の領域式は「1001＊」である．各内部節点の子の順序は，領域式に関するある大小関係（文献[3]で定義されているもの）に従っている．

外接長方形： 内部節点の各スロットでは，対応する領域中に中心点をもつすべての線分を含み，かつ各座標軸に平行な辺をもつ最小

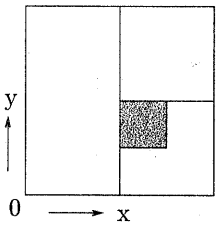
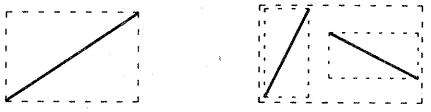


図 2: 領域式「1001∗」



(a) 葉のスロット　(b) 内部節点のスロット

図 3: 外接長方形の例

の長方形を保持している．また，葉の各スロットでは，1本の線分の外接長方形をもっている．例として，図 3(a) に葉のスロットの場合を，同図 (b) に内部節点のスロットの場合を示す．

以上のように，GBD 木では，線分データの管理を領域式と外接長方形によって行なっている．GBD 木に対する操作としては，データの挿入，削除，および範囲検索（対象平面内のある矩形範囲が指定されたとき，それと重なるすべてのデータを求める操作）が可能である．これらのうち，データの挿入，削除を行う際には，根から出発して，領域式を参照しながら子をたどっていくことにより，そのデータを記録すべき（あるいは，そのデータが存在している）葉に到達できる．また範囲検索時には，外接長方形を参照することにより効率よくデータを検索することができる．各操作の詳細については，文献 [3] を参照されたい．

2.2 改良 GBD 木の概略

文献 [4] では，GBD 木において長い線分が含まれる場合について考察している．例えば，図 4のように長い線分 1 が含まれているものとする．ここで，各領域 A~G について，外接長方形を破線で表している．今，図中に示すような検索範囲 W が与えられたとすると，W は葉 A, B の両方の外接長方形と重なりをもつので，両方の葉をたどらなければならない．このように，長い線分が含まれている場合，兄弟節点の外接長方形の重なりが多くなるため，検索範囲が与えられたときに無駄な節点をたどることがあり，一般に検索効率が低下する．

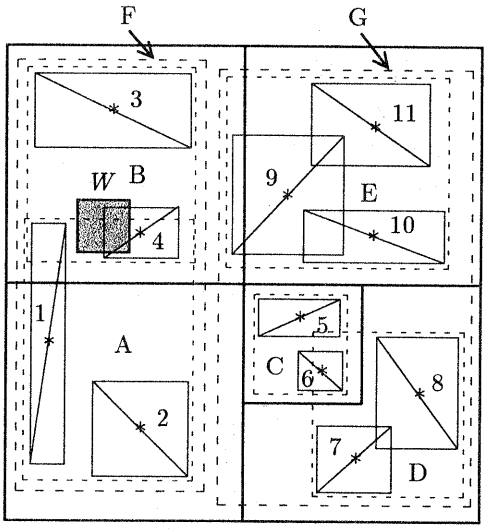


図 4: 長い線分が含まれる場合

この問題に対処するために，改良 GBD 木では，線分の外接長方形を，各辺の長さがある閾値 D_{max} をこえないように分割してサブ外接長方形を生成し，線分の外接長方形の代わりにこのサブ外接長方形を用いて GBD 木を構成している．すなわち，ある線分の外接長方形の横の辺の長さを L_x としたとき，それを $K_x \triangleq \lceil L_x/D_{max} \rceil$ 個に，縦の辺の長さを L_y としたとき，それを $K_y \triangleq \lceil L_y/D_{max} \rceil$ 個にそれぞれ分割して，合計 $K_x \cdot K_y$ 個のサブ外接長方形を生成する（$\lceil a \rceil$ は a より小さくない最小の整数を表す）．さらに，生成したサブ外接長方形のうち，線分の一部を含むもの（図 5の灰色の部分）のみを木に登録する．このようにして生成・登録したサブ外接長方形を，1本の線分データの外接長方形と同様に扱って改良 GBD 木を構成する．ここで，サブ外接長方形を格納した葉には，元の線分を示すポインタをもたせる．

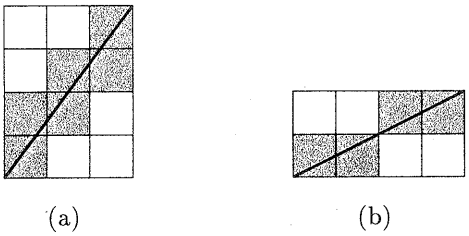


図 5: 登録されるサブ外接長方形

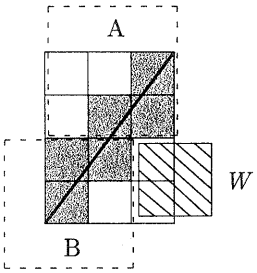


図 6: 分割の効果

このように，線分の一部を含まないサブ外接長方形を除外することにより，検索時に無駄な節点をたどることが少なくなる．例えば図 6 において，6 つのサブ外接長方形のうち，上の 3 つがある葉 A に，下の 3 つが別の葉 B にそれぞれ登録されるものとする．A, B の外接長方形および検索範囲 W が図中に示すようなものであるとすると，W に対する検索では，A, B いずれの葉もたどる必要がない．

さらに改良 GBD 木では，各線分の（サブ）外接長方形の一辺の長さが D_{max} 以下になるので，各節点の領域から外接長方形があまり大きくはみ出さなくなり，兄弟節点の外接長方形の重なりが少なくなる．これらの理由により，改良 GBD 木では GBD 木に比べて検索効率が向上している．

以上の分割法を図 4 の例に適用すると，図 7 に示すように，長い線分 1 に対して 1a, 1b の 2 つのサブ外接長方形が作られる．これにより，検索範囲 W が与えられたときに，葉 A をたどることなく線分データ 4 を検索できることになる．

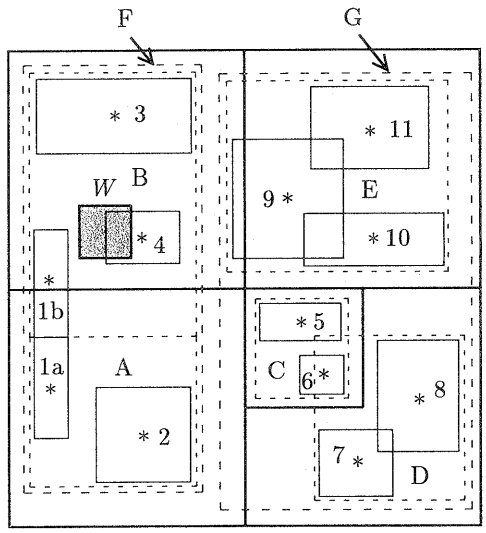


図 7: 改良 GBD 木による領域分割の例

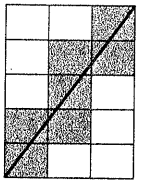
3. 改良 GBD 木の問題点と新しい分割方法

改良 GBD 木の問題点として，長い線分に対しては複数のサブ外接長方形が登録されるために，記憶領域が増えることがある．また，外接長方形の分割方法は非常に単純なものであり，まだ改善の余地が残されている．例えば，それぞれのサブ外接長方形が，必ずしも線分の各部分に外接する最小の長方形にならないことは，その部分を格納している葉の先祖の外接長方形を不必要に大きくし，検索効率を悪くしている可能性がある．

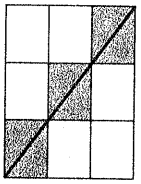
本章では，まず新しい 3 つの分割法を示す．これらの分割法 1~3 および後述する分割法 4 では，改良 GBD 木の分割方法（以下，従来法と呼ぶ）と同様，生成したサブ外接長方形のうち，線分の一部を含むもののみを木に登録する．

(1) **分割法 1**
従来法により外接長方形の短辺が分割される数 $K_{min} \triangleq \min\{K_x, K_y\}$ を求め，短辺，長辺ともに K_{min} 個に分割する．図 8(b) に例を示す．

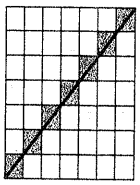
(2) **分割法 2**
従来法により外接長方形を分割したときに，登録されるサブ外接長方形の数 R を



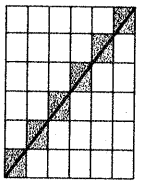
(a) 従来法



(b) 分割法 1



(c) 分割法 2



(d) 分割法 3

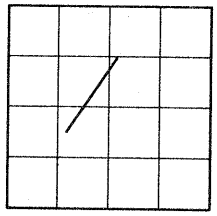
図 8: 分割法 1，2，3 の実行例

求める．そして，改めてもとの外接長方形を座標軸に平行な方向にそれぞれ R 等分する．図 8(c) に例を示す．

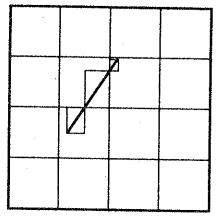
(3) **分割法 3**
従来法により外接長方形の短辺，長辺が分割される数 K_{min}, K_{max} を求める．そして，短辺，長辺ともに $K_{min} \times \lceil K_{max}/K_{min} \rceil$ 個に分割する．図 8(d) に例を示す．

分割法 1 は，単純にサブ外接長方形の数を減らすことによって，記憶領域の削減をねらっている．一方，分割法 2, 3 はサブ外接長方形の総面積を減らすものであり，兄弟節点の外接長方形の重なりを少なくし，検索効率を向上させることを目的としている．なお，いずれの方法でも，木に登録される各サブ外接長方形は，線分データが分割されてできるある部分に外接する最小の長方形となっている．

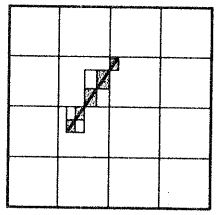
前述のように，各線分の（サブ）外接長方形の一辺の長さを D_{max} 以下にすることにより，各節点の領域からその外接長方形があまり大きくはみ出すことがなくなる．しかし，はみ出す距離が小さくても，大きな領域に対応する節点では，外接長方形の面積が大きく増加し，他の節点の外接長方形と重なる可能性が強くなる．次の分割法 4 は，このような状況を回避することによって，検索効率の改善を目指すものである（ただし，一般に，1 本の線分に対して生成されるサブ外接長方形の個数が，従来法に比べ多くなるため，記憶領域の増大を招くことになる）．



(a)



(b)



(c)

図 9: 分割法 4 の実行例

(4) **分割法 4**
線分データが，対象平面の各辺を 4 等分割する分割軸と交差するならば，まずこれらの分割軸により分割する（図 9(b)）．その後，分割された各部分について，その外接長方形が D_{max} 以上の長さの辺をもつならば，分割法 3 を適用する（図 9(c)）．

4. 計算機実験

4.1 実験方法

従来の改良 GBD 木と，分割法 1~4 を用いて構成される木を，以下のような計算機実験により比較した．

- 線分データ：対象平面は一辺の長さが 64 の正方形領域とし，線分の長さの上限を 40 に設定した．様々な長さ・位置・方向の線分 300 本からなるデータを，二次元疑似乱数によって 10 種類作成した．
- 木の構成手順：D_{max} の値は 4, 8, 16 の 3 通りを試した．従来法，分割法 1~4 のい

ずれについても，空の木から始めて，線分データを生成した順に挿入することにより木を作成した．なお，各節点のスロット数は 20 とした．

- 範囲検索：検索範囲は，一辺の長さが 1.28, 2.56, 3.84, 5.12 もしくは 6.40 の正方形領域とした．疑似乱数により，10 種類のデータのそれぞれに対し，それぞれの大きさの検索範囲を 10000 個ずつ生成した．

比較項目は以下の 2 つとした．

(i) 木の全節点数と葉の数
10 種類のデータに対する平均値を求めた．

(ii) 範囲検索時に訪問した節点数およびスロット数
それぞれの大きさの検索範囲について，$10 \times 10000 = 100000$ 回の検索に対する平均値を求めた．

項目 (i) は木の記憶領域の大きさを，(ii) は範囲検索効率を，それぞれ比較するためのものである．

4.2 実験結果

表 1~3 に実験結果を示す．

表 1 から分かるように，D_{max} の値が小さくなるにつれて木の節点数が多くなっている．これは，各線分に対して生成される（サブ）外接長方形の個数が多くなるためである．分割法 1 については，従来法に比べ木の節点数が大幅に減少しており，D_{max} の値が小さくなるほど，その減少の度合が大きくなっている．分割法 2 による木の節点数は従来法と同程度であり，分割法 3 では従来法より若干減少している．ただし，分割法 2 と 3 では大きな差はなく，特に $D_{max} = 16$ の場合にはまったく同じ木を生成している．分割法 4 では，対象平面の各辺を 4 等分する軸による分割が線分のサブ外接長方形を多くしているため，従来法に比べ木の総節点数が増加している．$D_{max} = 4$ の場合にはその度合は小さいが，$D_{max} = 16$ の場合には増加率が約 48% になっている．

範囲検索時の訪問節点数（表 2）および訪問スロット数（表 3）に関しては，分割法 1 による結

表 1: 節点数の比較

(a) $D_{max} = 4$

	従来法	分割法 1	分割法 2	分割法 3	分割法 4
全節点	168.0	65.1	169.3	163.5	168.9
葉	155.4	60.1	156.9	151.4	156.3

(b) $D_{max} = 8$

	従来法	分割法 1	分割法 2	分割法 3	分割法 4
全節点	85.7	38.6	85.9	85.4	99.7
葉	79.1	35.6	79.5	79.1	91.2

(c) $D_{max} = 16$

	従来法	分割法 1	分割法 2	分割法 3	分割法 4
全節点	45.5	29.0	44.1	44.1	67.4
葉	41.3	26.0	40.4	40.4	62.4

表 2: 訪問節点数の比較

(a) $D_{max} = 4$

検索範囲	従来法	分割法 1	分割法 2	分割法 3	分割法 4
1.28	5.60	8.84	5.12	5.13	4.38
2.56	6.78	9.81	6.21	6.21	5.35
3.84	8.13	10.84	7.47	7.44	6.49
5.12	9.71	11.94	8.95	8.89	7.82
6.40	11.55	13.09	10.69	10.59	9.43

(b) $D_{max} = 8$

検索範囲	従来法	分割法 1	分割法 2	分割法 3	分割法 4
1.28	6.16	9.16	5.64	5.64	4.74
2.56	7.06	9.92	6.46	6.46	5.53
3.84	8.06	10.70	7.39	7.38	6.44
5.12	9.17	11.51	8.43	8.42	7.49
6.40	10.35	12.34	9.57	9.56	8.64

(c) $D_{max} = 16$

検索範囲	従来法	分割法 1	分割法 2	分割法 3	分割法 4
1.28	6.86	9.53	6.47	6.47	5.35
2.56	7.52	10.18	7.10	7.10	6.01
3.84	8.23	10.86	7.77	7.77	6.75
5.12	8.97	11.50	8.46	8.46	7.51
6.40	9.83	12.22	9.28	9.28	8.42

表 3: 訪問スロット数の比較

(a) $D_{max} = 4$

検索範囲	従来法	分割法 1	分割法 2	分割法 3	分割法 4
1.28	77.35	118.4	70.83	70.92	60.34
2.56	94.03	132.1	86.31	86.22	74.08
3.84	113.1	146.7	104.0	103.6	90.29
5.12	135.5	162.0	124.9	124.2	109.4
6.40	161.5	178.3	149.4	148.2	132.2

(b) $D_{max} = 8$

検索範囲	従来法	分割法 1	分割法 2	分割法 3	分割法 4
1.28	80.25	123.9	73.31	73.46	58.69
2.56	92.82	134.7	84.91	85.03	69.33
3.84	106.8	145.9	97.89	97.94	81.64
5.12	122.4	157.3	112.6	112.5	95.67
6.40	138.9	169.1	128.6	128.5	111.4

(c) $D_{max} = 16$

検索範囲	従来法	分割法 1	分割法 2	分割法 3	分割法 4
1.28	85.88	119.2	84.62	84.62	66.45
2.56	95.04	128.1	93.58	93.58	75.65
3.84	104.9	137.4	103.2	103.2	85.86
5.12	115.2	146.0	113.1	113.1	96.37
6.40	127.3	155.9	124.9	124.9	109.0

果が，従来法に比べかなり悪くなっている．分割法 2, 3 では，木の総節点数が従来法とほぼ同じであるが，生成するサブ外接長方形を不必要に大きくしていないため，従来法に比べ訪問節点数が 5~8%，訪問スロット数が 1~8%程度改善されている．分割法 4 では，訪問節点数，訪問スロット数ともに，従来法に比べ 14~24%程度少なくなっている．対象平面の各辺を 4 等分する軸による分割が，兄弟節点の外接長方形の重なりを少なくし，検索効率の大きな改善につながったのだと考えられる．

5. むすび

改良 GBD 木の検索効率の向上を主な目的として，長い線分データの外接長方形の新しい分割方法を提案した．計算機実験を行ったところ，分割法 2, 3 では，範囲検索時の訪問節点数，訪問スロット数ともに，従来法より数%減少した．また分割法 4 では，木の総節点数は従来法より増加したものの，範囲検索時の訪問節点数および訪問スロット数は 14~24%程度減少した．

今後の課題は，記憶領域，検索効率の両方がより改善されるような木の構成法を見出すことである．現在，以下のような方法について検討を行っている．

- 木を構成した際に，同じ線分データに対する複数のサブ外接長方形が 1 つの葉に格納されることがあるが，それらを再び 1 つにまとめることにより，木の総節点数を減らす方法．
- 線分を分割するときに，サブ外接長方形の各辺の長さを D_{max} 以下にすることよりも，兄弟節点の外接長方形の重なりを大きくしないことを優先するような方法．

参考文献

[1] H. Samet : *The Design and Analysis of Spatial Data Structures*, Addison-Wesley, Reading, MA (1990).

[2] 中村 泰明, 阿部 茂, 大沢 裕, 坂内 正夫 : "空間的広がりをもつ図形データの MD 木による管理 —RMD 木—," 電子情報通信学会論文誌 (D-II), Vol.J73-D-II, No.12, pp.1976-1984 (1990).

[3] 大沢 裕, 坂内 正夫 : "2 種類の補助情報により検索と管理性能の向上を図った多次元データ構造の提案," 電子情報通信学会論文誌 (D-I), Vol.J74-D-I, No.8, pp.467-475 (1991).

[4] 下平 丕作士, 大沢 裕, 坂内 正夫 : "GBD 木の検索性能の改良方法 — 大きな図形を扱うための手法の提案 —," 情報処理学会論文誌, Vol.33, No.10, pp.1254-1262 (1992).

[5] 大沢 裕, 坂内 正夫 : "良好な動特性をもつ多次元点データ管理構造の一提案," 電子通信学会論文誌 (D), Vol.J66-D, No.10, pp.1193-1200 (1983).