# 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——

下平 丕作士†† 大沢 裕††† 坂内正夫††††

GBD木では，大きな図形が混在する場合，兄弟同士のノードの外接長方形の重なりが多くなるため，検索性能が低下する．この問題を解決するために，本論文では，図形の外接長方形を一定の大きさ（$D_{max}$）ごとに分割してサブ外接長方形を生成し，図形の外接長方形の代わりに，これらのサブ外接長方形を用いてGBD木を形成する方法を提案している．このとき，サブ外接長方形は実際に図形の一部を包含するもののみを登録する．この方法によれば，大きな図形の外接長方形は実質的に小さくなる．また，サブ外接長方形ごとに近くの図形とグループ化され，各ノードの外接長方形が小さくなり，兄弟同士の重なりが少なくなるため，検索性能が向上する．長い線分が混在するデータについての数値実験によると，適切な $D_{max}$ を用いることにより，提案方式では原方式に比べて検索時のCPUタイムとタッチノード数は低減されるが，挿入時のCPUタイムと木構造データを格納するのに要するメモリ量は増加することが分かった．削除時のCPUタイムには大きな差異はない．このような利点と欠点を考慮すると，提案した方法は実用上，長い線分が混在するデータを扱う場合で，検索性能を重視する場合に有用であると考えられる．

## 1. まえがき

近くに存在する図形同士を階層的にグループ化し，木構造で管理する図形データ管理構造の一種であるR木[1]やGBD木[2),3]は，動的に生成される高さがバランスした多分木構造であるため，大規模な図形データベースの管理に適しており，実用的な価値が高い．

図形データの管理においては，挿入・削除・検索処理を高速で行うためのインデックスが必要である．R木では，各図形について座標軸に平行で図形に外接する最もタイトな外接長方形を，また，図形同士をグループ化した各ノードについては，グループ内に含まれるすべての図形に外接する最もタイトな長方形を持っている．これらの外接長方形が，図形データのグループ化および検索のためのインデックスとして用いられる．この方式では，大きな図形が混在する場合には，兄弟同士のノードの外接長方形の重なりが生ずるため，検索時に一意に目的とする図形に到達できない場合がある．そこで，ノードの分割時に分割後の2つのノードの外接長方形ができるだけ小さくなるようにグループ化し，外接長方形の重なりを少なくするアルゴリズムを用いている．R⁺木[4]では，各レベルの空間分割において，各ノードの領域が重ならないように生成し，複数の領域にまたがる図形については，対応する各葉ノードに重複して図形データを持たせるようにしている．

GBD木では，図形同士をグループ化するためのインデックスとして領域式を，図形の検索のためのインデックスとしてR木と同様な外接長方形を用いている．ここで領域式とは，対象平面の2等分割を繰り返して得られる領域の位置と大きさを表すビット表現である．木構造データとして，葉ノードでは各図形の中心点（図心）の領域式を，また，非葉ノードではグループ内に存在するすべての図形の中心点を含む領域の領域式を持っている．挿入・削除時には，これらの領域式を用いることにより，近くに存在する図形同士のグループ化が，能率よく行われる．このため，GBD木の挿入・削除時の性能は，R木に比べて大幅に向上している．外接長方形の重なりについては特別な対策を講じていないが，図心の座標が近いもの同士がグループ化されるため，結果として近接する図形がグループ化されることとなる．

R木とGBD木の性能評価は，対象平面に比べて比較的小さい図形データによって行われている[1),2]．実用上は様々な分野のCADデータのように，小さい図形から大きな図形まで混在している場合が多い．大きな図形が混在する場合には，当然その外接長方形は大きく，したがって，これを包含する各ノードの外接長方形も大きくなり，相互の重なりも増えるため，検索

† A Method for Improving Retrieval Performance of a GBD Tree—A Proposal of a Method for Handling Large Graphic Elements—by HISASHI SHIMODAIRA (FM Service Department, Nihon MECCS Co. Ltd.), YUTAKA OHSAWA (Department of Information and Computer Science, Faculty of Engineering, Saitama University) and MASAO SAKAUCHI (Institute of Industrial Science, University of Tokyo).

†† 日本メックス(株)FM支援サービス事業部
††† 埼玉大学工学部情報工学科
†††† 東京大学生産技術研究所

性能が低下する．

外接長方形は，複雑な図形の形状を近似的に表現する最も簡単な方法であり，これによって図形の位置と広がりを表現できる．しかし，斜めの長い線分，凹多角形，穴のある図形等の外接長方形には，実際に図形が存在しない部分が多く含まれるため，検索性能の低下につながることとなる．

本論文では，大きな図形が混在する場合の GBD 木の検索性能の向上を図った改良方法を提案する．提案方式の基本的考え方は，大きな図形については，その外接長方形を一定の大きさごとに分割してサブ外接長方形を生成し，図形の外接長方形の代わりにサブ外接長方形を用いて GBD 木を形成することにある．図形を実際に分割するのではないため，各サブ外接長方形を格納した葉ノードのスロットに当該図形を指すポインタを持たせる．大きな図形の場合には，このサブ外接長方形によって検索が行われる．この方式により大きな図形の外接長方形を分割してサブ外接長方形を生成すると，もとの図形を含まないものが生成される場合があるが，あらかじめこれを除外することにより実質的に外接長方形が小さくなる．GBD 木の形成時には，大きな図形はサブ外接長方形ごとに近くの図形とグループ化され，各ノードの外接長方形が小さくなるため，結果として兄弟同士のノードの外接長方形同士の重なりが少なくなる．この2点により検索性能の向上を図っていることが，提案方式の特徴である．なお，提案方式の基本的な考え方は，R木等のように各ノードの領域の重なりを認めるタイプのデータ構造に適用できる一般的なものである．

以下本論文では，2章で GBD 木の改良方法の概要について述べる．3章では，提案した方式をインプリメントするためのアルゴリズムについて述べる．4章では，提案方式の各種性能を数値実験により検証する．なお，提案した方式は一般的図形に適用できるものであるが，本論文では，実用上最も頻繁に現れる大きな図形の代表として長い線分に適用し，その効果を確認している．

## 2. GBD 木とその改良方法の概要

### 2.1 GBD 木の概要

GBD 木では，ノードの各スロットに次のデータを持っている．かっこ内は，インプリメントする際に用いた値であり，1スロット当り合計25バイトである．

（1） 葉ノードでは図形の図心の領域式，非葉ノードではそのスロットが管理する下位ノードに存在するすべての図形の図心を包含する領域の領域式（4バイト）．

（2） 領域式の長さ（1バイト）．

（3） 外接長方形の左下隅および右上隅の $X, Y$ 座標（各4バイト，計16バイト）．

（4） 葉ノードでは図形の実データを指すポインタ，非葉ノードではそのスロットが管理する下位のノードを指すポインタ（4バイト）．

原方式[2]による GBD 木の例として，aからkまでの図形（○印は図心）が存在する場合の空間分割を図1(a)に，その木構造を図1(b)に示す．各ノードの領域は図1(a)の太い実線で，その外接長方形は点線で示すようになる．ここで，領域D（右下の4分の1の正方形）は図形hとiの図心を，領域E（右上の4分の1の正方形）は図形jとkの図心を包含している．各領域に対応するスロットは，それが管理するすべての図形を包含する外接長方形を持っている．この

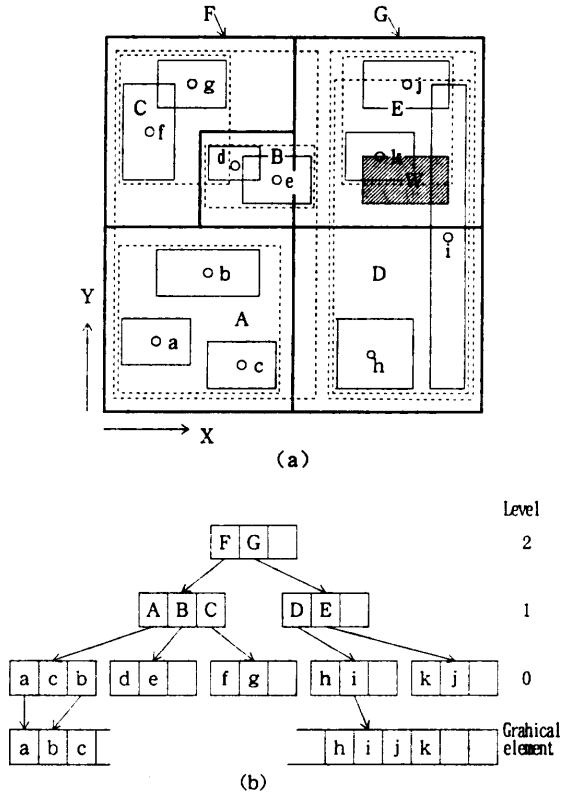


図 1 GBD 木の概要（原方式）
（a）空間分割，（b）木構造

Fig. 1 Outline of the GBD tree (Original method). (a) Spatial decomposition, (b) Tree structure.

場合，図形 i が大きいために，領域Dの外接長方形は，領域Eの外接長方形とかなりの部分重なっている．さて，検索範囲 W が与えられたとき，これと重なる図形 i と k にたどりつくには，領域 D と E の両方の子ノードを探索しなければならない（図1(b)参照）．このように，大きな図形が数多く存在するほど，兄弟同士のノードの外接長方形の重なりが多くなり，検索性能が低下することになる．

### 2.2 改良方法の概要

提案方式では，図形の外接長方形の辺長の限界値（$D_{max}$）を与え，辺長がこの値以上の外接長方形は，次のようにしてサブ外接長方形を生成する．外接長方形のいずれか一方の辺が $D_{max}$ より大きい場合（**図2**の(a)，(b)）には，長い方の辺を $D_{max}$ で割ったときの商（$K$）とその端数を考慮して，その辺を等分割してサブ外接長方形（図2では点線で表示）を生成する．図2(a)では $K=1$，(b)では $K=2$ であり，端数が存在するため，それぞれ $X$ 方向の辺を2等分割，$Y$ 方向の辺を3等分割する．両方の辺が $D_{max}$ より大きい場合（図2の(c)）には，各辺を $D_{max}$ で割ったときの商（$K_X$, $K_Y$）とその端数を考慮して，各辺を等分割してサブ外接長方形を生成する．図2の(c)では $K_X=1$，$K_Y=2$ であり，端数が存在するため，$X$ 方向を2等分割，$Y$ 方向を3等分割している．この場合には，さらに，サブ外接長方形が図形の一部を含むかどうかの判定を行って，登録の対象となるサブ外接長方形（**図3**のハッチングを施したもの）を選定する．このようにして生成・選定したサブ外接長方形を，1つの図形の外接長方形と同様に扱って GBD 木を形成する．各サブ外接長方形を格納した葉ノードのスロットには，もとの図形を指すポインタを持たせる．検索時には，これらのサブ外接長方形を用いてもとの図形を検索する．なお，提案方式をR木に適用する場合には，サブ外接長方形を図形の外接長方形と同様に扱って，その座標値に基づいて挿入やノードの分割を行えばよい．

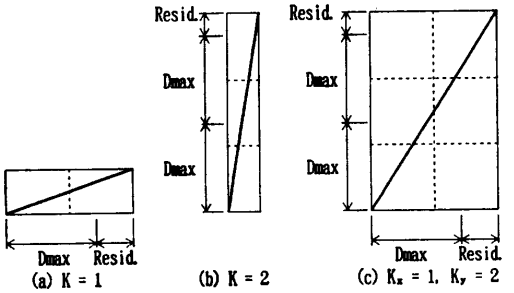


**図 2** 図形の外接長方形の分割
Fig. 2 Subdivision of minimal bounding rectangle of graphic element.

提案方式では，このように図形の一部を包含しないサブ外接長方形をあらかじめ除外するため，図3に示すように，斜めの線分や多角形の外接長方形を実質的に小さくすることができる．これにより，外接長方形の図形を含まない部分によって生じる無駄な検索パスを排除することができるため，検索性能が向上する．また，外接長方形の大きさが $D_{max}$ 以下にそろえられ，大きな図形については，サブ外接長方形ごとに近くの図形とグループ化される．その結果，各ノードの外接長方形の大きさが一定の大きさ以下になるため，兄弟同士の外接長方形の重なりが少なくなり，検索性能が向上する．提案方式の短所は，大きな図形については複数のサブ外接長方形が登録されるため，木構造データを格納するためのメモリ量と木の高さが増大することである．木の高さの増大は，その性能，殊に検索性能の低下につながる．したがって，実用に際しては，適用分野のデータの性能（対象平面と扱う図形の大きさの関係，大きな図形の量等）を考慮して，$D_{max}$ の値を選定することが必要である．

$R^+$ 木では，ノードの分割時に複数のノードの領域にまたがる図形については，各葉ノードにその図形データを重複して持たせるようにしているが，提案方式のように外接長方形の不要な部分を除外するという工夫は行われていない点で提案方式と異なっている．

図1と同一の例題に提案方式を適用すると，空間分割は**図4**(a)に，木構造は図4(b)に示すようになる．この場合には，図形 i については，図4(a)に示すようにサブ外接長方形 $i_1$ と $i_2$ が生成される．これらのサブ外接長方形を用いて，その図心の領域式を求めて GBD 木を生成する．その結果，i は $i_1$ と $i_2$ に分かれてグループ化されるため，領域Dは図形 h と $i_1$ を，領域E は $i_2$, j, k を包含することとなる．この場合，領域Dの外接長方形が小さくなるため，領域Dと領域

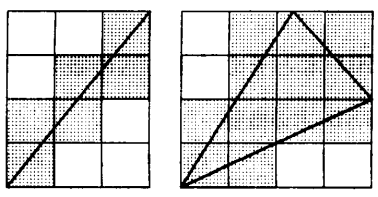


**図 3** 登録の対象となるサブ外接長方形
Fig. 3 Sub-bounding-rectangles to be registered.

Eの外接長方形の重なりは生じない．$i_1$ と $i_2$ は，ともに同一の図形 i の実データを指すポインタを持っている．この方式によれば，検索範囲Wが与えられたとき，領域Eの子ノードを探索するだけでこれと重なる図形 i と k を得ることができる（図4（b）参照）．

## 3. 提案方式のアルゴリズム

### 3.1 図形データの挿入

GBD 木への図形データの挿入は次の手順で行われる．

I1：挿入図形の外接長方形を求める．外接長方形の座標値を用いて，図形が与えられた限界値（$D_{max}$）以上であるかどうかを，判定する（図2参照）．

I1.1：各辺が $D_{max}$ より小さい場合，図形の図心の領域式を求め，I2 の処理に移る．

I1.2：いずれか一方の辺が $D_{max}$ より大きい場合（図2（a），（b）），2.2 節で述べた方法によってサブ外接長方形を生成し，その座標値と図心の領域式を求める．すべてのサブ外接長方形について処理した後，I2 の処理に移る．

I1.3：両方の辺が $D_{max}$ より大きい場合（図2（c）），2.2 節で述べた方法によってサブ外接長方形を生成し，頂点の座標値を求める．生成したサブ外接長方形について，そのサブ外接長方形が図形の一部を含むかどうかの判定を行って，登録の対象となるサブ外接長方形を選別する．登録の対象となるすべてのサブ外接長方形について図心の領域式を求めた後，I2の処理に移る．

以下の処理を，次の場合分けに従って行う．

（1） 外接長方形の分割が行われなかった場合，挿入する図形について I2，I3，I4 の処理を行う．

（2） 外接長方形の分割が行われた場合，登録の対象となるサブ外接長方形について，その個数だけ I2，I3，I4 を繰り返す．

I2：次の判定を行いつつ，木を根から葉に向けてたどる．

I2.1：葉以外のノードではスロットの領域式と挿入データの領域式との比較を最も左のスロットから行っていき，最初に見つかった挿入データを包含するスロットが示すノードをたどる．

I2.2：葉ノードに到達したとき，その葉ノードのスロットに空きがあればそこにデータを入れる．既に $N$ 個（ノードのスロット数）のデータが存在するときは，「葉ノードの分割」を行う．

I3：子ノードの分割が起こったとき，新たに作られたノードを親ノードに挿入する．その結果，親ノードにおいて子ノードの数が $N+1$ 個となったとき，「非葉ノードの分割」を行う．

I4：木構造を根ノードに向けて戻りつつ，外接長方形を更新する．

葉ノードや非葉ノードの分割は，サブ外接長方形を1つの図形の外接長方形と同様に扱って，原方式[2]のアルゴリズムによって行う．上記のアルゴリズムと原方式の主な差異は，サブ外接長方形の生成と挿入処理である．

### 3.2 図形データの削除

図形データの削除は，次の手順で行われる．

D1：削除する図形を指示する点の座標が与えられる．$k_P$ に根ノードをセットする．

D2：$k_P$ が葉ノードでないとき，各スロットの外接長方形に指示点が含まれるかどうかをチェックし，該当するスロットが管理する子ノードを $k_P$ にセットす

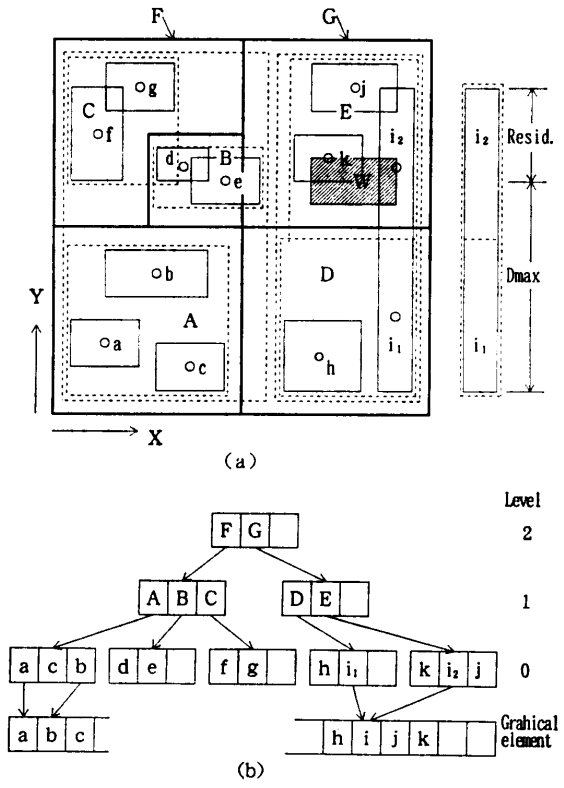


**図 4** 提案方式による GBD 木の概要
（a）空間分割，（b）木構造

Fig. 4 Outline of the GBD tree by proposed method.
(a) Spatial decomposition, (b) Tree structure.

る．この処理を再帰的に繰り返して，木を葉に向けてたどる．

D3：$k_P$ が葉ノードのとき，各スロットの外接長方形とその図形に指示点が含まれるかどうかをチェックし，該当する図形を見つける．

D3.1：該当する図形について，次の処理を行う．

（1） 外接長方形の各辺が $D_{max}$ より小さい場合，図形データを葉ノードから削除し，D4 の処理に移る．

（2） 外接長方形のいずれかまたは両方の辺が，$D_{max}$ より大きい場合，D3.2 の処理に移る．

D3.2：次のようにして，登録されているすべてのサブ外接長方形のデータを削除し，D4 の処理に移る．

D3.2.1：I1.2 と I1.3 と同様にして，登録されているサブ外接長方形の図心の領域式を求める．

D3.2.2：サブ外接長方形の領域式とノードの各スロットの領域式の包含関係を調べることにより，木を葉に向けてたどり，削除しようとするサブ外接長方形が置かれている葉ノードを見つける．

D3.2.3：領域式が一致するサブ外接長方形のデータを葉ノードから削除する．

D4：その結果，ノード中のデータ数が〔$(N+1)/3$〕未満となったとき，兄弟の葉ノードとの併合を行う．

D5：もし併合の結果，データ数が $N$ を越える場合は，「葉ノードの分割」を行う．

D6：木構造を根に向かってたどりつつ，下位ノードの併合の結果，使用スロット数が〔$(N+1)/3$〕未満となった非葉ノードに対し，非葉ノードの併合を行う．

ノードの併合は，サブ外接長方形を1つの図形の外接長方形として扱って，原方式[2]のアルゴリズムによって行う．上記のアルゴリズムと原方式の主な差異は，D3.2 におけるサブ外接長方形の削除処理である．

### 3.3 図形データの検索

長方形の検索範囲 W が与えられたとき，非葉ノードの各スロットの外接長方形と W との重畳関係のチェックを行いつつ，木を葉ノードに向けてたどる．葉ノードでは，外接長方形と W の重畳関係のチェックを行い，ついで，図形自身と W との重畳関係のチェックを行って重なる図形を見つける．提案方式では，大きな図形の検索は，サブ外接長方形をインデックスとして行われる．

## 4. 性能テスト

### 4.1 テスト内容およびテスト方法

GBD 木の原方式と提案方式をインプリメントし，数値実験により各種性能の評価を行った．使用した計算機は，NEC PC-9801 RA5 (80387 数値演算コプロセッサを使用）であり，MS-DOS Ver. 3.30C の環境下で，C 言語により，図形の実データと木構造のデータをすべて主記憶に格納するようにインプリメントした．

提案方式では，対象平面に比べて比較的大きな図形が混在するデータを対象としている．実際の応用面では，長い線分を含むデータが多いため，ここでは，線分データによって提案方式の一般的性能をテストした．線分の生成にあたっては，長さの最大値 $L$ を設定し，一様疑似乱数によって様々な長さ・位置・方向の線分が混在するデータを作成した．線分が存在する対象平面を辺長が 64 の正方形領域とした．

最初に，適切なノードのスロット数を選定するために，スロット数を変化させてテストを行い，その結果に基づいて，以後のテストで用いるスロット数を 20 スロット（ノードサイズ 500 バイト）とした．

テストは，**表1**に示す3シリーズについて行った．テストシリーズ1は，データサイズ（線分数 $M$）が一定の場合の最大線分長と $D_{max}$ の大きさの関係を調べるためのものである．このテスト結果に基づいて，テストシリーズ2と3では，$D_{max}=16$ を用いることとした．テストシリーズ2は，データサイズが一定の場合の最大線分長の影響を調べるためのものである．テストシリーズ3は，最大線分長が一定の場合のデータ

**表 1** テストデータの内容とテスト方法
Table 1 Contents of test data and applied method.

| No. | Contents of data | Applied methods | Results |
|---|---|---|---|
| 1. | $M=3000$, $L=40$, 60, Change $D_{max}$ : 8, 16, 24, 32 | O. M. and P. M. | Fig. 6 |
| 2. | $M=3000$, Change $L$ : 10, 20, 30, 40, 50, 60 | O. M. and P. M. ($D_{max}=16$) | Fig. 7 |
| 3. | $L=60$, Change $M$ : 1000, 2000, 3000, 4000 | O. M. and P. M. ($D_{max}=16$) | Fig. 8 |

O. M. : Original Method, P. M. : Proposed Method.

サイズの影響を調べるためのものである．

$M$=3000, $L$=40 のデータの線分の分布を，図5に示す．実用上は，3000 個の図形データを含むデータは，たとえば，建築図面データではかなり大きなデータに相当する．$L$=60 の場合の最大の外接長方形の辺長は，ほぼ対象平面の辺長に等しい．また，$L$=40 の場合の最大の外接長方形の辺長は対象平面の辺長の約 60% に等しい．$L$=40 で $M$=3000 の場合，線分の外接長方形の平均面積の総和（付録参照）は，対象平面の面積の約 97.7 倍であり，原方式によれば多量の非葉ノードの重なりが生ずる．$D_{max}$=8, 16, 24, 32 は，外接長方形の辺長の限界値を対象平面の辺長の約 1/8, 1/4, 1/2.7, 1/2 にとることを意味する．このとき，たとえば，$L$=40 で $D_{max}$=16 の場合には，分割の対象になる外接長方形は 16 から 40 までの長さの線分の外接長方形であり，全体のたかだか 60% にあたる．

テストは文献 1) と同様な方法で行った．まず，あらかじめ主記憶に読み込んでおいた線分データによって GBD 木を構築し，その際の残りの 10% の線分をランダムに挿入する際の CPU タイムを計測した．ついで，長方形の検索範囲により，全体の 5% の線分を検索する際の CPU タイムを計測した．最後に，全体の 10% の線分をランダムに削除する際の CPU タイムを計測した．

### 4.2 テスト結果

各テストシリーズについて，挿入・削除・検索処理時の1図形（提案方式による検索では，サブ外接長方

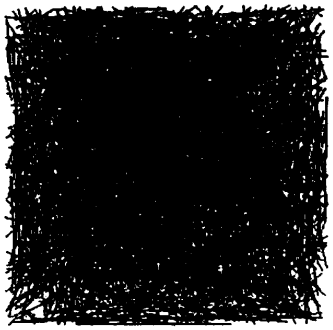

図 5 線分の分布状況 ($M$=3000, $L$=40)
Fig. 5 Distribution of line segments ($M$=3000, $L$=40).

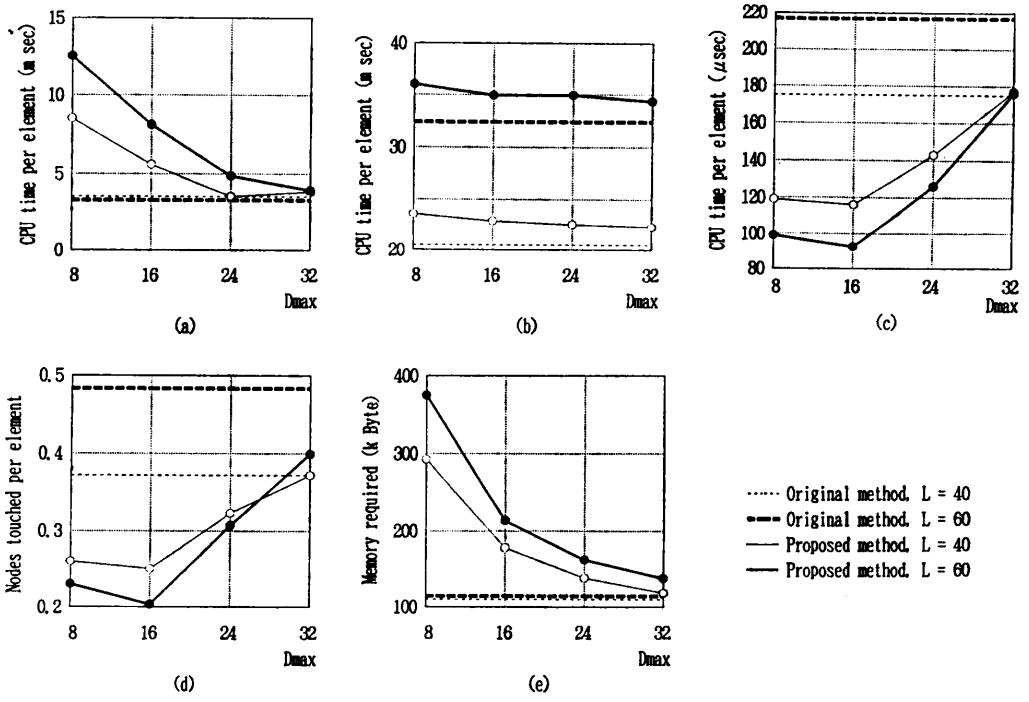


図 6 $D_{max}$ と性能の関係
(a) 挿入時の CPU タイム，(b) 削除時の CPU タイム，(c) 検索時の CPU タイム，
(d) 検索時のタッチノード数，(e) 木構造データ所要メモリ量
Fig. 6 Relation between $D_{max}$ and the performance.
(a) CPU time for insertion, (b) CPU time for deletion, (c) CPU time for search,
(d) Nodes touched for search, (e) Memory required for tree data.

形ではなく本来の1図形）当りのCPUタイム，検索時の1図形当りのタッチノード数（検索時にタッチしたノード数を該当図形数で除したもの），および木構造データについての所要メモリ量を，図6から図8に示す．テスト結果の主な特徴は，次のとおりである．

テストシリーズ1（図6）：提案方式によると，線分が長い場合，$D_{max}$ を小さくすると検索性能が向上し，$L=60$ で $D_{max}=16$ の場合に，検索時のCPUタイムとタッチノード数は原方式に比べて約1/2.4になっている．一方で，挿入時の処理時間は $D_{max}$ を小さくすると増加し，$D_{max}=16$ の場合に，原方式の約2.3倍になり，所要メモリ量も約1.9倍になっている．$D_{max}=8$ の場合に検索性能が低下しているのは，木の高さの増大によるものである．削除時の処理時間については，提案方式の方が若干大きいが，それほど大きな差はない．

テストシリーズ2（図7）：原方式では，線分長は挿入時の処理時間と所要メモリ量には大きな影響を及ぼさないが，削除時と検索時には線分が長いほど処理時間がかかる．提案方式では，原方式に比べて線分が長いほど検索性能向上の効果が大きくなるが，一方で挿入時のCPUタイムと所要メモリ量は増加する．

テストシリーズ3（図8）：原方式と提案方式を比較すると，挿入時の両者の処理時間についてはデータサイズによる影響は少ないが，データサイズが大きくなると，提案方式の検索時のCPUタイムとタッチノード数の改善の程度が大きくなる傾向が見られる．また，削除時の両者の処理時間の差は少なくなっている．メモリ量については，データサイズにかかわらず提案方式は原方式の約1.9倍である．

### 4.3 提案方式の評価と有用性

提案方式は，線分以外の多角形等の複雑な形状の図形にも適用できる．原方式と提案方式における図形の形状による処理の相違は，図形の外接長方形の計算と検索時の検索範囲と図形が重なるかどうかのチェックであり，複雑な形状の図形については線分よりも処理時間が増える．しかし，両方式はともに同じようにその影響を受けるため，両者の性能の差は線分データの場合と同様であり，したがって，提案方式を複雑な形状の図形を含むデータに適用した場合にも，線分データと同様な効果が得られるものと考えられる．しかし，実験結果からは線分以外のデータに適用した場合

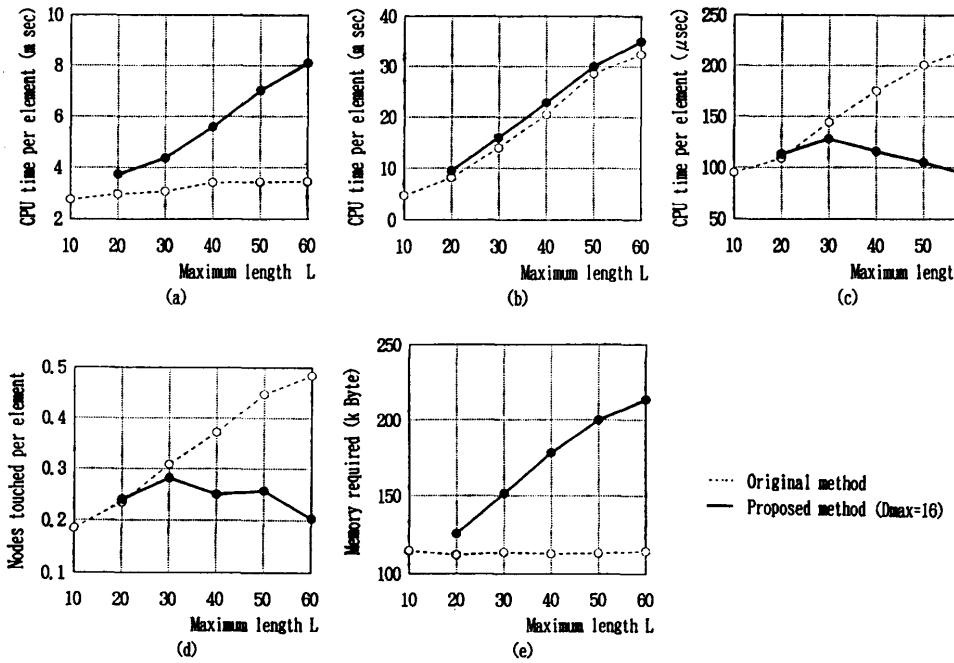


図7 最大線分長と性能の関係
（a）挿入時のCPUタイム，（b）削除時のCPUタイム，（c）検索時のCPUタイム，
（d）検索時のタッチノード数，（e）木構造所要メモリ量
Fig. 7 Relation between maximum length and the performance.
(a) CPU time for insertion, (b) CPU time for deletion, (c) CPU time for search,
(d) Nodes touched for search, (e) Memory required for tree data.

の性能については確認できないため，以下の提案方式の有用性についての議論は，大きな図形として長い線分が混在するデータへの適用に限定する．

提案方式では，対象平面に比べて長い線分が混在する場合に，適切な $D_{max}$ を用いることにより，原方式に比べて検索時の CPU タイムとタッチノード数が低減される．また，提案方式は，$D_{max}$ の値により挿入・削除・検索の性能をコントロールできるという特徴がある．実験結果は，応用分野のデータの性質によって，検索性能を向上させるのに最も適切な $D_{max}$ の値が存在することを示唆している．一方，提案方式は原方式に比べて，挿入時の処理時間と木構造についての所要メモリ量が増加するという欠点がある．

他の手法と比較すると，原方式の GBD 木の検索性能は，R 木と同等かそれ以上である[2]．類似の方式の $R^+$ 木では，R 木に比べて検索性能を含めてその性能に決定的な差異はなく，しかも，アルゴリズムが複雑でコーディングが難しい[5]．上記の結果から，提案方式は検索性能が優れており，しかも，$R^+$ 木のように複雑なアルゴリズムを用いないでも，比較的簡単に GBD 木に組み込める点に特徴があると言えよう．

提案した方式は，上記のように利点とともに欠点もあるため，実用上の有用性は用途や利用環境に依存するが，次のような場合に提案方式の利点を生かすことができる．

（1） 建築図面は，対象平面の辺長と同程度の長さの基準線とスパン長（基準線間の距離）の数倍から数分の一の寸法の壁や柱等の部品から構成されている．したがって，提案方式は，建築図面データを扱うのに適している．

（2） CRT 画面上で挿入・削除する図形データを入力する対話型 CAD では，通常，これらの一回の操作の対象となる図形の個数は数個程度であるのに対し，画面の部分拡大操作の際には数十から数百の図形の検索が頻繁に行われる．このとき，一回の操作の処理時間は検索時の方が挿入・削除時よりもオーダが大きくなるため，挿入性能が低下しても検索性能の向上のメリットが大きい．

（3） 提案方式では，検索時の 1 図形当りのタッチノード数は原方式に比べて大幅に減少するため，木構造データまで 2 次記憶に置かなければならない利用環境の場合で，検索性能を重視する場合に有用である．

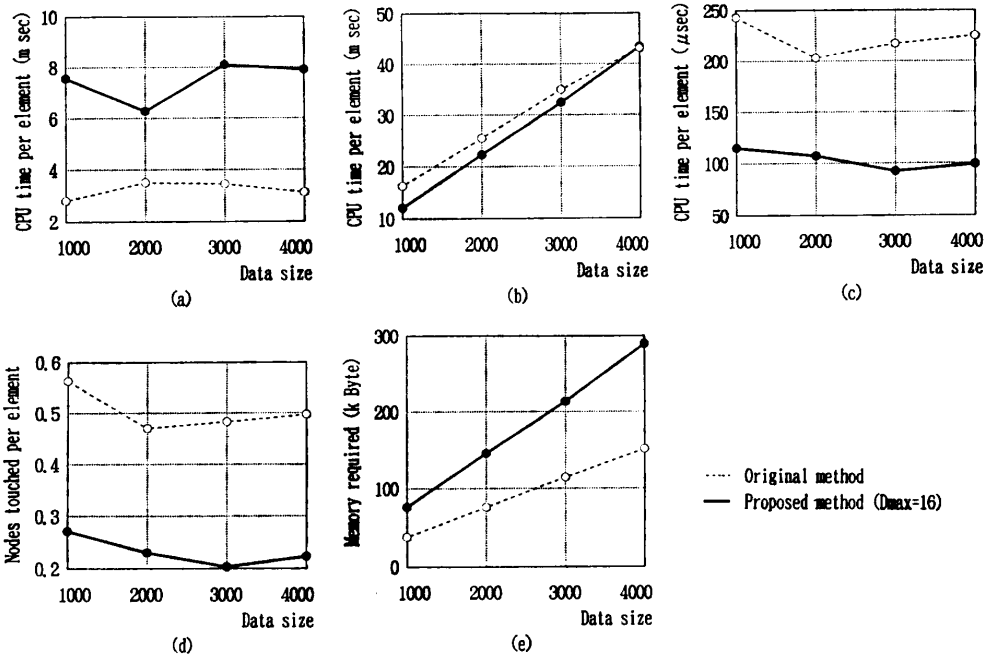


図 8 データサイズと性能の関係
（a）挿入時の CPU タイム，（b）削除時の CPU タイム，（c）検索時の CPU タイム，
（d）検索時のタッチノード数，（e）木構造データ所要メモリ量
Fig. 8 Relation between date size and the performance.
(a) CPU time for insertion, (b) CPU time for deletion, (c) CPU time for search,
(d) Nodes touched for search, (e) Memory required for tree data.

## 5. むすび

本論文では，GBD 木について，大きな図形が混在する場合に外接長方形の重なりをできるだけ少なくなるようにして，検索性能の向上を図った改良方式を提案した．長い線分が混在するデータについての数値実験より，次のことが分かった．適切な値の $D_{max}$ を用いることにより，提案方式は原方式に比べて検索時の CPU タイムとタッチノード数が低減されるが，挿入時の CPU タイムと木構造データについての所要メモリ量は増加する．削除時の処理時間は原方式と比べて大きな差はなく，データサイズが大きくなるとほとんど差がなくなる．線分データについての原方式に対する提案方式の主な性能比は，範囲検索時の1図形当りの処理時間とタッチノード数については最大で1/2.4，挿入時の処理時間については2.3倍，木構造データについての所要メモリ量については1.9倍である．

上記のような利点と欠点を考慮すると，提案した方式は実用上，長い線分が混在するデータを扱う場合で，検索性能を重視する場合に有用であると考えられる．

実際の複雑な形状の図形を含むデータに適用して，その性能を検証することが今後の課題である．提案方式の考え方をR木に適用した場合にも，GBD 木と同様な効果が期待できるため，今後検討する予定である．

**謝辞** 本研究は，著者の一人がNTTに在職中に行ったものである．研究の機会を与えていただいた関係者の方々に感謝の意を表します．

## 参考文献

1) Guttman, A.: R-trees: A Dynamic Index Structure for Spatial Searching, *Proc. of ACM SIGMOD*, Vol. 14, No. 2, pp. 47-57 (1984).
2) 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), Vol. J74-D-I, No. 8, pp. 467-475 (1991).
3) Osawa, Y. and Sakauchi, M.: A New Type Data Structure with Homogeneous Nodes Suitable for a Very Large Spatial Database, *Proc. of 6th International Conference on Data Engineering*, pp. 296-303 (Feb. 1990).
4) Sellis, T. et al.: The R+ Tree: A Dynamic Index for Multi-dimensional Object, *Proc. of 13th VLDB Conference*, pp. 507-518 (1987).
5) Greene, D.: An Implementation and Performance Analysis of Spatial Data Access Methods, *Proc. of 5th International Conference on Data Engineering*, pp. 606-615 (1989).

## 付　録

線分の外接長方形の平均面積の総和 $S$ は，次のようにして求められる．長さ $x$ の線分の外接長方形は，線分のなす角度が 45° の場合に最大値 $x^2/2$ であり，0° または 90° の場合に最小値0である．したがって，その平均面積は $x^2/4$ である．0から $L$ までの間に $M$ 本の線分が一様に分布するものとすると，

$$S=\int_0^L \frac{x^2}{4}\frac{M}{L}\mathrm{d}x=\frac{M}{4L}\int_0^L x^2\mathrm{d}x=\frac{1}{12}ML^2$$

（平成4年2月6日受付）
（平成4年7月10日採録）

**下平丞作士**（正会員）

昭和16年生．昭和44年東京都立大学工学部建築工学科卒業．昭和46年同大学院修士課程修了．同年日本電信電話公社（現株式会社）入社．武蔵野電気通信研究所および建築部建築技術開発室において，数値解析・CAD・データベース・AI などの基礎理論とシステム開発に従事．平成4年から日本メックス(株)において，建物・設備管理用システムの研究・開発に従事．CAD，建物・設備管理用システムなどへの AI 技術の応用に興味を持っている．工学博士．日本建築学会，電子情報通信学会各会員．

**大沢　裕**（正会員）

昭和 51 年信州大学工学部電子卒業．昭和 53 年同大大学院修士課程修了．東京大学生産技術研究所助手等を経て，現在，埼玉大学工学部情報助教授．図面自動認識，地理情報処理，多次元データ管理構造の研究に従事．工学博士．

**坂内　正夫**（正会員）

昭和 44 年東京大学工学部電気卒業．昭和 50 年同大大学院博士課程修了．同年東京大学工学部電気専任講師．昭和 51 年横浜国立大学工学部情報助教授．昭和 53 年東京大学生産技術研究所助教授．昭和 63 年同教授，現在に至る．マルチメディアデータベース，図面・画像などからのデータベース取得，パターンデータ構造，画像処理，地理情報処理，信頼性などの研究に従事．著書「VLSI コンピュータII」（岩波書店，共著），「画像データベース」（昭晃堂，共著）など．工学博士．