重複性の導入による図形検索構造の性能改善方法

2S－4

下平　丞作士

日本メックス株式会社

１．はじめに

Ｒ木[1]やGBD 木[2]では，図形に外接する最小の長方形を索引として用いて検索する．これらの手法では，外接長方形相互の重なりを認めており，重なっている部分では探索パスが一意ではない．したがって，長い線分等大きな図形を扱う場合，外接長方形相互の重なりが増加するため，検索性能が低下する．本報告では，大きな図形を扱う際の検索性能の改良方式について報告する．

２．改良方式の概要

改良方式では図形の外接長方形の辺長の限界値(Dmax)を与え，図形の外接長方形の辺長がこの値以上のときは，サブ外接長方形を生成し，もとの外接長方形の代わりに用いる．サブ外接長方形は，図１に示すようにもとの外接長方形の各辺をDmaxで区切ったときの数と端数を考慮して，各辺を等分割して生成する．このとき，サブ外接長方形のうち，図形の一部を包含するもののみを登録する（図２）．大きな図形については，図形の挿入時にこのようにしてサブ外接長方形を生成し，その個数分だけ挿入処理を行う．この際，Ｒ木ではサブ外接長方形の座標を，GBD木ではその図心の領域式を用いて挿入処理を行う．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ーを持たせる．検索時には，サブ外接長方形を用いてもとの図形を検索する．この方式によれば，大きな図形の外接長方形は実質的に小さくなる場合が多く，木の形成時にはサブ外接長方形毎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に，結果として外接長方形相互の重なりが少なくなり，検索性能が向上する．

長い線分Ｂがある場合のＲ木の例を，図３に示す．線分Ｂの外接長方形が大きいため，ノードR1とR2の外接長方形はかなり重なっている．この例に改良方式を適用すると，図４のようになる．線分Ｂの外接長方形は，B1からB4のサブ外接長方形に分割され，ノードR1,R2,R3に分かれてグループ化されるため，R1,R2,R3の外接長方形の重なりは生じない．検索範囲Wが与えられたとき，該当する図形ＢとＤを得るために，もとのＲ木ではR1とR2を探索し，ＡからＥ

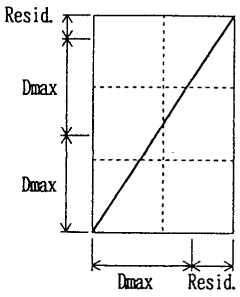


Fig.1 Way of subdividing bounding-rectangle

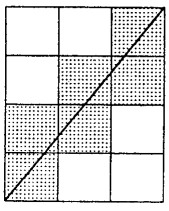
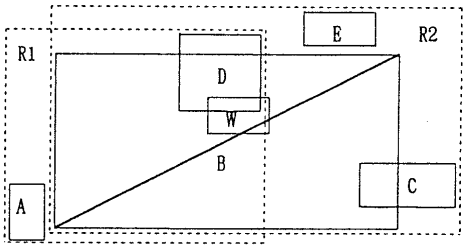
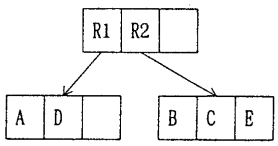


Fig.2 Sub-bounding-rectangles to be registered

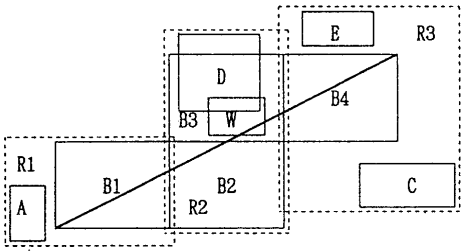


(a) Rectangles organized into an R-tree

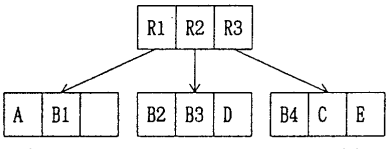


(b) R-tree for the rectangles of (a)

Fig.3 Example of original R-tree



(a) Rectangles organized into an R-tree



(b) R-tree for the rectangles of (a)

Fig.4 Example of R-tree employing the proposed method

Method for Improving the Performance of Spatial Index Structures by Introducing Redundancy

Hisashi Shimodaira

Nihon MECCS Company Ltd.

までの図形をチェックしなければならないが，改良方式ではR2のみを探索し，B2,B3,Dの図形をチェックすればよい．

3．性能テスト

GBD 木について原方式と改良方式をインプリメントし，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全て主記憶に格納するようにインプリメントした．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CAD データについてGBD 木を構築し，その際の残りの10%の図形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%の図形を検索する際のCPU タイムを計測した．最後に，全体の10%の図形をランダムに削除する際のCPU タイムを計測した．

基準線やバルコニー等長い線分を含む建築平面図データを用いて，性能テストを行なった．表１にテストの内容を示す．シリーズA1では，図５に示すデータについて，Dmaxを変えてテストを行なった．この建物は，X方向14スパン（スパン長７m），Y方向８スパン（スパン長７m）である．図形数は1000個である．Dmaxは，スパン長の約1/2,1,2,4,7 倍である．シリーズA2では，Dmaxを８mとして，図５のデータの角度を変化させたものについてテストを行なった．テスト結果を，表２と表３に示す．CPU タイムとタッチノード数は，１図形当り（改良方式ではサブ外接長方形ではなく，本来の１図形）である．

4．むすび

改良方式をGBD 木に適用した結果，次のことが分かった．改良方式では，長い線分については，スパン長程度の寸法で外接長方形を分割すれば，若干の所要メモリ量が増えるが，削除と検索の性能がかなり向上する．挿入時の処理時間は増える場合も減る場合もある．

参考文献

1)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)

2)Guttman A. : R-trees: a dynamic index structure for spatial searching, Proc. of ACM SIGMOD, Vol.14, No.2, pp.47-57 (1984)

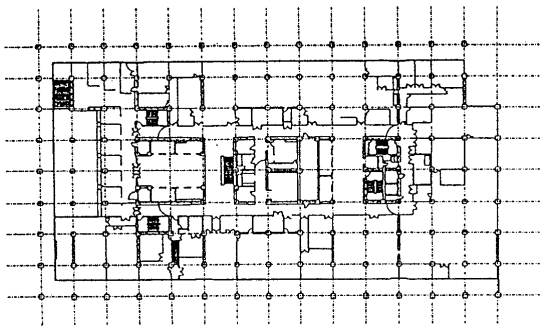


Fig.5 Architectural plan drawing data

Table 1 Test Configuration of architectural CAD data

No		Contents of data	Applied method	Results
A1	M=1000	Change Dmax: 4,8,16,32,48 m	O.M. and P.M.	Table 2
A2	M=1000	Change Angle: 0,15,30,45 deg.	O.M. and P.M. (Dmax=8m)	Table 3

O.M. : Original method　P.M. : Proposed method

Table 2 Relation between Dmax and performance (Test A1)

Measured item	Original method	Prposed method Dmax: 4	8	16	32	48
CPU time per object for insert. (m sec.)	3.40	4.40	2.70	2.40	3.52	3.34
CPU time per object for delet. (m sec.)	5.96	5.24	5.44	5.48	5.98	5.58
CPU time per object for search (μsec.)	140	113	83	98	99	122
Nodes touched per object for search	0.306	0.245	0.163	0.204	0.204	0.265
Memory required for tree data (k Byte)	45.9	68.4	52.7	48.3	47.4	45.4

Table 3 Relation between data angle and performance (Test A2)

Data angle	CPU time per object for insert. (m sec.)		CPU time per object for delet. (m sec.)		CPU time per object for search (μsec.)		Nodes touched per object for search		Memory required for tree data (k Byte)	
(degree)	O.M.	P.M.	O.M.	P.M.	O.M.	P.M.	O.M.	P.M.	O.M.	P.M.
0	3.40	2.70	5.96	5.44	140	83	0.306	0.163	45.9	52.7
15	3.36	2.52	8.30	4.68	166	132	0.340	0.255	45.4	53.2
30	3.22	3.58	8.60	4.68	213	142	0.435	0.283	44.9	57.6
45	3.72	3.88	5.28	5.26	233	155	0.500	0.326	43.9	58.6