構造的データ表現による空間インデックスの効率化

堀之口，浩征
九州大学大学院システム情報科学研究科知能システム学専攻 ： 博士後期課程

牧之内，顕文
九州大学大学院システム情報科学研究科知能システム学専攻

https://doi.org/10.15017/1513735

出版情報：九州大学大学院システム情報科学紀要. 5 (1), pp.87-92, 2000-03-24. 九州大学大学院システム情報科学研究院
バージョン：
権利関係：

九州大学大学院
システム情報科学研究科報告
第5巻　第1号　平成12年3月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.5, No.1, March 2000

構造的データ表現による空間インデックスの効率化

堀之口 浩征* ・ 牧之内 顕文**

Improving the Performance of Spatial Index Based on Structured Data Representation

Hiroyuki HORINOKUCHI and Akifumi MAKINOUCHI

(Received December 10, 1999)

Abstract: The spatial index takes very important role on accelerating the spatial query processing. The index structure for the spatial space is made from two processes. The first one called filtering uses abstract figure and extracts candidates which may involve user wanted data quickly. And the next, refinement distills those candidate using exact data representation taken from the databases. The refinement process needs very large I/O, so removing unnecessary candidates which do not meet the user's requirements is the key to gain better performance. At first we point out the problem of R*-tree which prevents the index from filtering more candidates off. Then, we propose new technique to operate on that problem and show its performance with simulations.

Keywords: Structured data representation, Spatial index, Filtering, Refinement

1. まえがき

空間データベースとは空間属性を持つデータを効率的に管理をする構造である．空間データベースの具体的応用は，ITSにおける地理情報の利用や，情報携帯端末による発信者の位置特定，鯨等の無線による追跡等の空間的な位置や関係に着目した分野に広がっている．

空間インデックスはこれらの空間データベース内に蓄えられた膨大なデータの中から利用者に必要なデータを効率的に取り出す助けとなる仕組みであり，データベースの現実的な応用において欠かせない．空間インデックスの対象となる空間データは領域的な広がりを持つために，その特徴に応じた構造が求められる．

今回の空間インデックスにおける研究は，我々が現在研究を進めている時空間データベースシステムHawks[13),14)]における，インデックス実現の足掛かりである．そのため，比較的十分な研究成果が得られている空間インデックスを採用する．一方で，この従来の構造をそのまま利用しただけでは，システムの利点である柔軟なデータ表現を生かす事が出来ない事も分かった．

そこで，本手法では，空間インデックスに構造的な領域表現を持たせ，空間データベースに対する検索効率を向上させる事を提案する．空間データの管理構造構築時に領域に関する情報を導入し，木構造の改良を試みた．結果として，空間領域の特徴に応じた本検索手法の有効性を確認した．

本論文の構成は以下のとおりである．まず，2.章において，空間インデックスとして何が必要であるか，従来の手法での問題点は何かについて論じる．3.章では，従来の空間検索の構造と手法をどの様に拡張するのか，また，その利点と欠点について紹介する．4.章では，計算機実験により実際の効果を調べ評価する．5.章では，空間インデックスの他の実現方法等関連する研究について説明し，最後に6.章で今回のインデックスについてまとめる．

2. 空間データを扱うインデックス

空間データベースの応用例としては，NASAのEOSDISプロジェクト†1があげられる．同プロジェクトでは，地球科学のための膨大なデータを空間データベースにより管理し，このデータを効率的に利用する手法を研究している．Sequoia 2000 project[6)]のベンチマークもその1つで，川や湖等のオブジェクトの位置を数値化し，その形状をPoint，Raster，Polygon，Graphで表現している．これらの数種の形式で表わされるデータを管理し，問合せを効率良く処理するために空間インデックスが用いられる．

2.1 空間データとその近似形状

空間インデックスでは，内部のデータ構造の操作を容易にするために，対象のオブジェクトを近似形状で代用する．MBR（Minimum Bounding Rectangle）は，これらの近似形状の1つであり，対象オブジェクト（群）を包

平成11年12月10日受付

* 知能システム学専攻博士後期課程
** 知能システム学専攻

†1 http://eospso.gsfc.nasa.gov/eos_homepage/eosdis.html

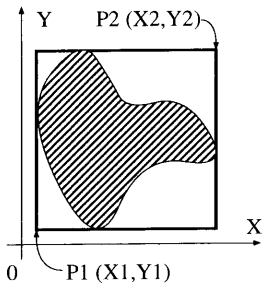

Fig.1　A MBR

含する最小の矩形で，かつ各辺が対応する空間軸に平行な領域である．具体的には，二次元のMBRは長方形であり，3次元では直方体となる．同時に，次元数に非依存に空間内の二点により表現出来る．

MBRの二次元の例として，**Fig. 1**をあげる．ここで，斜線領域で表されるオブジェクトのMBRは，オブジェクトを囲む太い線の矩形である．そして，このMBRはP_1 *(X1,Y1)*とP_2 *(X2,Y2)*の2点で表される．もし対象が3次元のオブジェクトならば，3次元の一対の点P_1 *(X1,Y1,Z1)*およびP_2 *(X2,Y2,Z2)*の形で表せる．同様にして，PolygonやGraph等のオブジェクトもその表現形式を問わず，2点のみで近似される．MBRの利用によりオブジェクトの位置管理の負担を小さく出来る．

MBRの取り扱いにおいては，n次元MBRを直接に用いる方法とそれを2n次元の領域等に変換して扱う方法の2つがあげられる[5]．たとえば，上記の2次元のP_1とP_2とで表されるMBRは4次元の点(X1, Y1, X2, Y2)で表すことが可能である．MBR間には交差領域が存在するが，点間には存在しない．そのために点集合にはMBR集合に比べ，より単純にクラスタリングができる利点がある．一方で，空間インデックスに対する問合せの1つである最近接検索では，直接MBRを用いる方が優れているとの報告も出ている[8]．また，2n次元で表現した場合には，2n次元内の遠近関係は現実の遠近を反映出来ない．我々は，直接MBRを用いる方法を選び，その中で広い応用範囲で良い性能を示しているR*-treeを用いる．直接MBRを用いない方法や，空間インデックスの他の実現方法については5.章で説明する．

2.2　近似に基づく質問処理

R*-treeは近似形状であるMBRを階層的に管理する木構造である．そのため，その出力はデータベースの領域検索の出力と一致しない．つまり，MBRを用いた出力集合の中には，すべての答に加えて答ではないものも含まれる．R*-treeの現実的な応用においては，出力の各オブジェクトに対して，問合せの条件を正確に満たすかどうかを検査しなければならない．

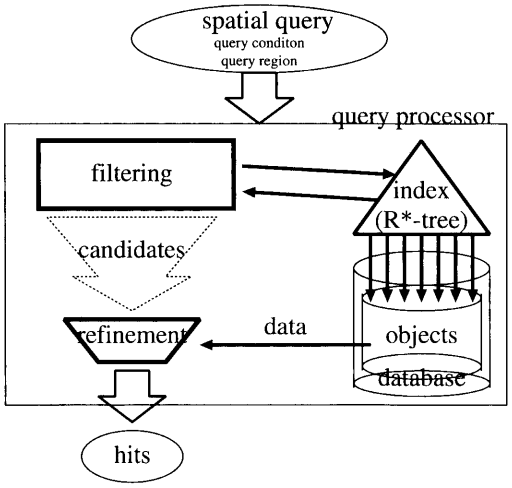

Fig.2　The work flow of query processing.

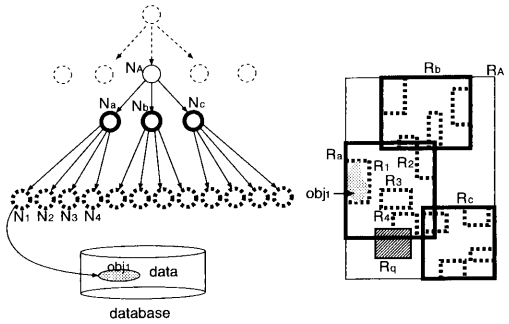

Fig.3　The structure of R*-tree. Each node in left tree has a region. Node N_n(n=A, a-c or 1-4) has a corresponding region R_n.

空間インデックスは，**近似に基づく質問処理**[10]に基づいて構成される．その略図を**Fig. 2**に示す．R*-treeは**Fig. 2**中におけるフィルタリング（filtering）において用いられる．フィルタリングの出力を受けるリファインメント（refinement）では，データベースからオブジェクトを取得したうえで，それが持つ正確な空間属性値が問合せの（空間に関する）条件を満たすか否かを判断する．データベースからのオブジェクト取得には大量のdisk I/Oが伴う．加えて，問い合わせ条件を満足するか否かの判定には計算機幾何学に基づく計算が必要となるため，リファインメントは一般に重い処理である．そのため，フィルタで条件を満足しないオブジェクトを可能な限り排除することが，空間データベース全体の効率的観点から重要となる．

2.2.1　R*-tree

空間インデックスでは領域的な広がりを持つデータを対象とする．幅を持つ物体群を対象にしたインデックスとしてR-tree[4]が有名であり，R*-treeをはじめ，いくつか類似の構造[11),7),9)]を派生させている．

R*-tree やR-treeでは，すべてのオブジェクトのMBRを多分木の構造で管理する．各ノードは1つのMBRと複数の子ノードへのポインタを持つ．葉ノードに限っては子ノードを持たないが，代わりにデータベース内のオブジェクト自身（のレコード）への参照を持つ．また，葉ノードが持つMBRは，対応するオブジェクトのMBRであり，中間ノードには自分の各子ノードのMBRの集合を包含するMBRがある．つまり，ノードの親子関係が，MBR間の包含関係と対応している．**Fig. 3**は二次元空間のオブジェクトに対するR*-tree の構成の例である．図中において，丸は木のノードを表わす．図の左側のR*-treeが右側のMBRの包含関係と対応する．常に子の領域を親の領域が包含しているので，親の領域が質問領域と交差を持たないならば，子の領域と質問領域は交差を持ち得ない．又，1つのノードが持つことのできる子の数には上限と下限が決められ，あるノードが上限に達した時点でそのノードは分割される．これにより，木は常に完全平衡木の状態を保つ．

以上はR-treeとR*-tree に共通した特徴である．R-treeは構造を作る時点で空間質問処理の効率が決まる。つまり、新しく投入するノードを木の中でどこに位置付けるかが質問処理時の効率を左右する。新しいノードのより良い接続先を決める方法は複数考えられるが、これらの方法は排他的であるために、どれが良いかの比較は難しい。R*-treeではこのノード接続先の選択方法がR-treeと異なる。

2.2.2 リファインメントにおける交差判定処理

リファインメントでは，オブジェクトの具体的なデータ表現に依存した交差判定の処理が必要である．今回我々が対象としているのはSequoia 2000 project[6]のベンチマークデータの形式，即ち，オブジェクトをその表面上の点列によって表現する形式である．

リファインメントにおける交差判定処理で最低限必要となる処理は，領域に対する点の内外判定である．多角形に対する与点の内外判定として，winding number method[1]と呼ばれている本法が有名であるので，今回はこれを比較対象として用いた．winding number methodでは，無限遠の点から与点に対して半直線を延ばした時に，多角形表面(線分，若しくは点)のどの部分に，どの様に，そして，何度交わるかで判定する．この方法では，複雑な多角形に対しても適用出来る代わりに，多角形表面上の線分の数だけ線分の交差判定を繰り返す必要がある．また、半直線が多角形の頂点と交差する時には、交差後に領域内部に入るか、そのまま領域外部へ抜けるかを調べなければならない。

3. 空間インデックスの拡張

点若しくは矩形との交差を調べるために，空間インデックスでは二段階で処理を行う．先ず，一段階目(フィルタリング)においては，空間データのMBR情報のみを用いて粗い判定を行なう．その上で，二段階目(リファインメント)では，先の段階で得られた結果を精査する．一段階目に比べ，二段階目の処理コストは大きい．そこで我々は，コストと処理内容の点から，一段目と二段目の中間に入る様な構造とそれを用いた処理方法を提案する．

Fig.4 Gains more performance by advanced filtering.

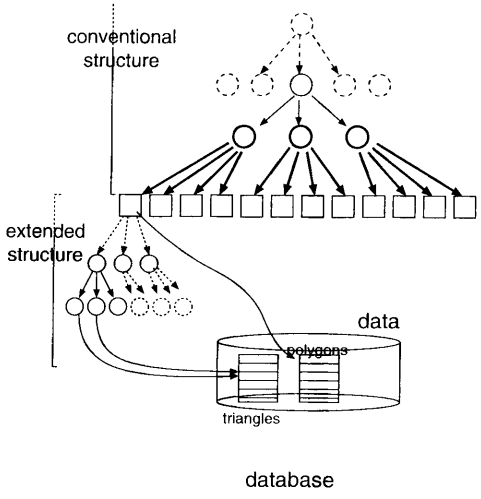

Fig.5 The structure of extended filtering.

3.1 空間内オブジェクトの管理構造

二段階の検索を組み合わせる事で，総合的なコストを低減させるのが，空間インデックスの要件である．フィルタリングの主眼は，オブジェクトのおおまかな位置を少ないデータ量で管理する点にある．又，どの様なMBRの管理構造を用いるにせよ，この段階での出力からは，**このオブジェクトは与領域(点を含む)と交差する可能性がある**以上の情報は得られない．我々は，リファインメントでの入出力等の処理コストを考えた上で，**このオブジェクトのこの部分に領域の交差の可能性がある**のレベルの情報をリファインメント部分に伝える手段について検討した．この情報伝達は，リファインメントの処理の一部を省く可能性を生じさせる．

3.2 本構造の特徴

今回提案する構造の基本的構成を**Fig. 5**に示す．木を用いる点では従来と変わらない．また，**Fig. 5**において，*conventional structure*とある部分の構成や構築法に関し

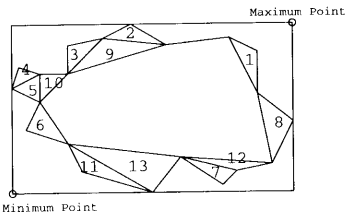

Fig.6　Cutting triangles around the polygon.

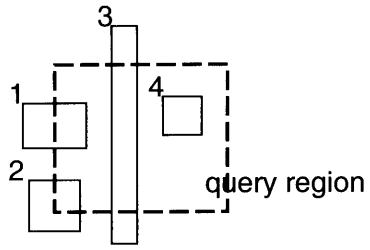

Fig.7　Figures which have intersection like 1, 2 and 4 have no need to test intersection.

ても同じである．**Fig. 3**との大きな違いは木の葉が持つデータにある．従来の構造における葉の直接の参照先は，具体的なオブジェクト形状表現に関するデータであった．今回はこれに代わり，オブジェクトのおおよその形状を表現する構造を葉に持たせる．オブジェクト形状を表現する構造は次の二つからなる．

- オブジェクトの内部に含まれる凸多角形
- オブジェクトの内部で，上の凸多角形に含まれない領域を構成する三角形の集合

Fig. 6にある様に、多角形の頂点を巡回しつつ、三角形を分離する事で、多角形から一つの凸多角形と複数の三角形を作る。このうち，三角形の集合の方は木(具体的にはR*-tree)を用いて管理する．多角形の表面に近い部分を三角形で管理するのは、空間質問の交差判定の時に外側にある部分をより細く管理するためである。

3.2.1　新構造における検索方法

Fig. 5における*conventional structure*とある部分の検索方法は従来と同じである．与えられた領域と交差する可能性のあるノードを選択し，それ以下のノードを全数調べていく．葉に到った場合には凸多角形，及び三角形集合との与領域の交差を調べる．

3.2.2　本構造の応用の適否

本構造の特徴は，フィルタリングの段階でオブジェクトの空間内に占める位置に加え，その形状に関するデータを持たせる点にある．そのため，表現力が増した分だけのデータ量の増加は避けられない．

この構造が有効になると考えられるのは，オブジェクトの形状が複雑な場合である．ここでの複雑な形状とは，オブジェクトの面が多数の線分により構成されている場合を指している．リファインメントの要は，オブジェクト表面上の全ての線分に対する交差判定である．今回の構造は，オブジェクト内部を表現する凸領域を持つため，この部分に関する判定には，交差判定が必要な部分を二分探索を用いて限定出来る．加えて，三角形集合についても，その領域の小ささが交差判定時に有利に働く可能性がある．

検索処理においては，与領域とオブジェクトの交差，包含の関係が調べられる．この時，領域同士の交差を判定する場合には，MBRの交差の型のみを見るだけで十分な場合が存在する．この状況を図で説明したのが**Fig. 7**である．MBRの中にどんな形状が含まれていようとも，1,3,4の各領域(細い実線の矩形)は与領域(破線の矩形)と交差を持つ．2については，MBRだけでは判定出来ず，実際の形状を調べなければならない．MBRが小さければそれだけ包含される可能性が高まり，このMBRに着目した比較が意味を持つ．

また，今回の構造が利用出来るのは，オブジェクトが表面上の点列によって表現されている場合のみに限られない．この構造が持つのはオブジェクトの内部領域であるので，例え，オブジェクトが曲線で保持されていたとしてもフィルタリングでの判定において有効に機能する．

空間的な広がりを持つデータを扱うのが今回の構造の特徴ではあるが，幅を持つどの様なデータに対しても有効であるとは限らない．この例として，離散的に変化する状態を表現するデータ，例えば，従業員の給与額推移等の形状が単純なデータを挙げる事が出来る．階段状に変化していくデータでは，今回のインデックスの特徴である複雑な形状に対する有効性を発揮する事が難しくなる．時間推移に主眼を置いたインデックス構造[12]も既に存在する．2.節にて挙げた二段構造の有効性についても検討も必要になると思われる．

Table 1　実験環境

OS	FreeBSD 3.3-STABLE
CPU	Celeron(334.09-MHz)
DISK	QUANTUM FIREBALL EX6.4A
real memory	131072K bytes
avail memory	125024K bytes
swap	267664K bytes

4.　評　　価

今回は，本構造を用いた領域検索の効率について実験により検証した．実験環境を**Table–1**に示．実験の基となるデータとして，現実の地理情報データである，Sequoia 2000 projectのベンチマークデータの一つ，Polygonを用いた．Sequoiaでは，一つのオブジェクトは，

表面上の点座標のリストの形で表現される．そのため，今回提案している手法はリファインメントの処理の一部をフィルタリング時に行なう事に近い．ただ，実際の質問処理においては，フィルタリング処理時に交差判定を行なうのか，フィルタリングの結果をリストの形で一旦作った後にまとめて交差判定を行なうのかとの違いがある．

オブジェクトを分割して得られる，点，三角形及び木(R*-tree)の各データ要素の集合は各々別々のファイルとして管理され，プログラム実行時にメモリ空間にマップされる．これにより，各データは永続性を得ている．

4.1 実験内容

本実験では，オブジェクト集合に対して，winding number methodと本構造を用いて空間インデックスを構築した上で領域検索を行なう．性能比較の基準は，検索の処理時間及びI/O(page fault数)である．また，検索に与える領域として点を用いる．領域として点を用いるのは，点による検索は他の領域検索(矩形検索や最近接検索)の基となる重要な判定処理であるのが理由である．与える点はオブジェクト空間内にランダムに生成した上で用いた．

まず，リファインメントの有無で出力数にどれだけの差があるかについて調べる．これにより，検索処理におけるリファインメントのコストの高さを見る．次に，各々の構造に対して独立に，構造の構築と検索を実行する．検索対象となるオブジェクトの数は固定である．今回のFreeBSDシステムでは，インデックスのデータ(約45Mbyte)がフィルタリングの初期の段階でキャシッングされてしまうので，今回の比較実験では計算部分の比較の要素が強く出た可能性は否めない．一方で，現実的な環境に近くなっていると言える．検索プログラムは与えられた領域と交差するオブジェクトの検索を1024回刻みに8セッション行なう．検索プログラムの起動は各構造に対して30回行なった．その上で，より平均的値を得るために上位と下位から各々3回分の値を除いた．

4.2 実験結果と考察

リファインメントの有無による処理コストの違いを示すのが，**Fig. 8**である．図では検索プログラムの領域質問の処理数とその処理にかかった時間(user time)の関係を表わす．図で判別出来る一本の線がリファインメントである。他方のフィルタリングは0.0に近い所にあるために、図からは判別不可能である。出力データからフィルタリングはリファインメントの1/70程度の時間しか必要としていない事が分かった．この事からリファイメントはフィルタリングに比べて格段の処理コストが求められている．なお，今回の検索プログラムにおいてはsystem timeを必要とする処理は少なく，比較の対象となり得なかったため，ここでは省いている．

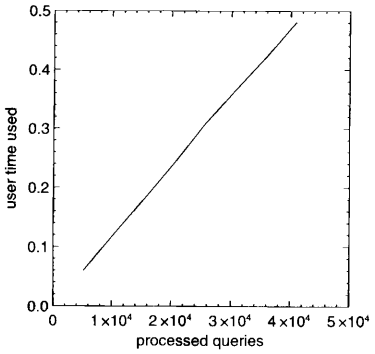

Fig.8 Requirement for the refinement.

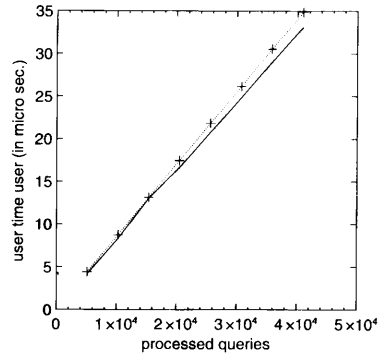

Fig.9 Structured object representation vs. winding number method.

次に**Fig. 9**において、winding number method(破線のグラフ) と今回の構造(実線のグラフ)の比較を行なっている。比較の条件は**Fig. 8**と同様に，領域質問の処理数とその処理時間である。今回の新しい構造を用いる事で，winding number methodを使った方法に比べ，約5%の処理コスト低減に成功した．

5. 関 連 研 究

空間インデックスにも様々な実現法や応用がある．いずれにしても，高速かつ不正確なフィルタリングと低速かつ正確なリファインメントの併用が基本的に用いられている．フィルタリング，若しくはリファインメントの性能向上を目的としたオブジェクト表現のインデックスへの導入は幾つか試みられてきた．空間インデックスの効率改善を目的とした他のオブジェクト表現方法と我々の用いた構造とを比較する．

5.1 オブジェクト表現の拡張

Asanoらが導入したのは，個々のオブジェクトを水平方向にスライスし，多数の台形を生成した上で，R-treeを用いて管理する方法[3]である．この方法では単一方向に多数(台形)の分散が生じる．多数のオブジェクトの断片が

存在する条件はR*-treeのクラスタリングを妨げる方考に働く可能性がある．個々のオブジェクトを三角形の集合として表現する方法はPreparataらにより提案されている[2]．この場合には，オブジェクトを分割した上でリファインメントの処理も行なう点で我々の研究との類似がある．個々の要素を効率的に管理出来る条件下においては，有効と考えられるのは我々と同じ条件である．また，生成される要素の数は我々の方法よりも多くなる可能性がある．

5.2 他の空間インデックスの応用

時間や空間にとらわれず，より高次元の空間を対象としたインデックスとして，空間インデックスを応用した例がいくつか存在している．X-tree[9]はR*-treeをベースにしている点や，時間や空間の属性にとらわれない点で我々の方法と共通する．X-treeでは，中間ノードの持つ子ノード数（fanout）が可変であるノード（Supernode）を用意している．これに加え，中間ノード間の領域の重複をなくしている．しかし，次元が数十から百程度の領域を対象としている点が我々のアプローチと異なる．

この構造に正規化を導入すれば，我々の場合と同様に性能向上が期待できると思われるが，これは今後の課題である．

6. むすび

空間インデックスにおける質問処理を効率化するために，空間データの表現と管理構造に対して，どの部分が性能向上の妨げになっいるか，又，どの様な改良の余地が残されているかについて論じた．空間データに対する新たな応用が広がりつつある現在，空間データベースにおいて不可避な検索コストをどう低減するかは重要な問題である．

本論文で提案する空間インデックスでは，すでに一般的となっている二段階の処理方法の上に構築する事が可能である．つまり，既存のインデックスデータを一から作り直す事は必ずしも必要とはならない事を意味している．また，今回のオブジェクト表現の構造化はインデックス構築時により多くの情報を保持する事で，領域検索の処理コストを低減している．

今回の構造の性能改善に与える影響について，現実の地理データに対する計算機実験により確かめた．その上で，従来の交差判定の処理コストを5%程度減らせる事を確認した．点領域の質問処理は空間インデックスの現実的応用における基本部分であり，その有効性に関しては他の領域質問に対しても通用すると考えられる．今回得られたデータを基に，今後はこの他の領域質問に関する有効性を検討していく予定である．

参考文献

1) *COMPUTER GRAPHICS, A Programming Approach, Second Edition.* McGraw-Hill Book Company, 1987.
2) Preparata F.P., Shamos M.I. Computational Geometry Springer, 1985.
3) Asano Ta., Asano Te.. Minimum Partition o Polygonal Regions into Trapezoids In Proc. 24th IEEE Annual Symp. on Foundations of Computer Science, pages 233-241, 1983.
4) A. Guttman. R-trees: a dynamic index structure for spatial searching. In *Proc. ACM SIGMOD Int. Conf. on Management of Data*, pages 47–57, 1984.
5) Jack Orenstein. A comparison of spatial query processing techniques for native and parameter spaces. In *Proceedings of the 1990 ACM SIGMOD*, pages 343–352, 1990.
6) Michael Stonebraker, Jim Frew, Kenn Gardels, and Jeff Meredith. The sequoia 2000 storage benchmark. In *Proceedings of the 1993 ACM SIGMOD*, pages 2–11, 1993.
7) N. Beckmann, H. Peter Kriegel, R. Schneider, and B. Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD*, pages 322–331, 1990.
8) Norio Katayama and Shin'ichi Satoh. The SR-tree: An index structure for high-dimensional nearest neighbor queries. In *ACM SIGMOD '97 5/96 Tucson, Arizona*, pages 369–380, 1997.
9) Stefan Berchtold, Daniel A. Keim, and Hans-Peter Kriegel. The X-tree: An index structure for high-dimensional data. In *Proceedings at the 22nd VLDB Conference Mumbai(Bombay), India*, pages 28–39, 1996.
10) T. Brinkhoff, H. P. Kriegel, and R. Schneider. Comparison of approximations of complex objects used for approximation-based query processing in spatial database systems. In *Ninth International Conference on DATA ENGINEERING*, pages 40–49, 1993.
11) Timos Sellis, Nick Roussopoulos, and Christos Faloutsos. The R$^+$-tree: A dynamic index for multi-dimensional objects. In *Proceedings of the 13th VLDB Conference, Bringhton*, 1987.
12) 天笠 俊之, 有次 正義, 田中 貴之, 金森 吉成. 時空間概念データモデルの実装. 情報処理学会論文誌：データベース, Vol. 40, SIG 6 (TOD3), pages 141–151, 1997.
13) 堀之口 浩征, 黒木 進, 牧之内 顕文. 時空間データモデルuniverseを用いた空間インデックスの実装と評価. 第4回電子情報通信学会九州支部学生会講演会 講演論文集, page 79. 電子情報通信学会・九州支部, 1996.
14) 堀之口 浩征, 黒木 進, 牧之内 顕文. 時空間データベースインデックス正規化r*-treeの実装と性能テスト. 情報処理学会 論文誌, 1225–1235. 情報処理学会, 1999.