九州大学学術情報リポジトリ
Kyushu University Institutional Repository

新しい自己組織化マップに基づくR*-tree ： 構築と検索

馮, 尭鍇
九州大学大学院システム情報科学府知能システム学専攻 ： 博士後期課程

久保, 正明
九州大学大学院システム情報科学府知能システム学専攻 ： 修士課程

アグバリ, ザヘル
九州大学大学院システム情報科学研究院知能システム学部門

牧之内, 顕文
九州大学大学院システム情報科学研究院知能システム学部門

https://doi.org/10.15017/1515738

出版情報：九州大学大学院システム情報科学紀要. 6 (2), pp.209-214, 2001-09-26. 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：

KYUSHU UNIVERSITY

九州大学大学院
システム情報科学紀要
第6巻 第2号 平成13年9月

Research Reports on Information Science and
Electrical Engineering of Kyushu University
Vol.6, No.2, September 2001

新しい自己組織化マップに基づくR*-tree: 構築と検索

Yaokai FENG* ・久保正明** ・Zaher AGHBARI***・牧之内顕文***

A New SOM-based R*-tree: Building and Retrieving

Yaokai FENG, Masaaki KUBO, Zaher AGHBARI and Akifumi MAKINOUCHI

(Received June 14, 2001)

Abstract: R-trees are widely used in spatial and multi-dimensional databases. However, according to our investigation, the overlap among the leaf nodes of R-trees is serious and the objects are not well-clustered in the leaf nodes, which greatly affect the effect of the pruning strategies when nearest neighbor searching is performed and also affect the other search performance of R-trees. The forced reinsertion introduced in R*-tree can improve this problem to some extent, but can not completely solve this problem. In this study, we try to combine SOM (Self Organizing Map) technology and R*-tree technology to lessen the overlap among the leaf nodes of R*-tree and to improve the clustering degree of the objects in the leaf nodes. The experimental result shows that the SOM-based R*-tree proposed in this paper has a much better search performance than R*-tree.

Keywords: Self-organizing maps, R*-tree, Multi-dimensional index, Nearest neighbor search

1. はじめに

マルチメディアデータに大きく依存するコンピュータアプリケーションの増加から, 近年, データベース分野ではマルチメディアデータの処理, 検索に注目が集まっている. しかし, マルチメディアデータは, 普通多次元空間での特徴ベクトルを用いて識別するため, インデックス作成にはポイントアクセスメソッド(PAM:point access method)の問題が生じる.

例えば, 画像はカラー, テクスチャー, そして輪郭などの特徴ベクトルで表現される. 特徴ベクトルとは対象の性質を表したものであり, 通常高次元になる. 例えば, $n_1 \times n_2$ピクセルを持つ画像のカラー特徴ベクトルは$3 \times n_1 \times n_2$次元で表される. 係数3は, 輝度, 色差情報に沿ったYIQ空間(NTSC方式で利用されている) [1]でのそれぞれの画素が持つ値の数である. つまり, 画像は特徴ベクトルにより$3 \times n_1 \times n_2$次元空間での一点で表現される.

現在, R-treeファミリー [2], 特にその中でも最も有名なR*-tree[3]はよく利用される多次元インデックスの階層構造である. しかし, 我々の研究ではR-treeの葉ノードの重なりがひどく, オブジェクトもうまくクラスタリングされてない. これは, 非常に大きな問題であり, 検索性能を低下させている. 特に, 片寄って分散したデータの時は低下が著しい. 我々の研究によるとR-treeでの葉ノードの重なりとクラスタリングが問題である理由は次の通りである.

1. R-treesのクラスタリング機能が強固ではない.
   R-treeやR*-treeを構築すれば葉ノードにはオブジェクトがクラスタリングされる. しかし, R-treeもR*-treeもノードにどれだけのエントリを確保するかは決まっていない. 違った挿入順序であれば違った木が構築される. 状況次第ではその状況下で優れた検索性能を保証できないノードを生成する可能性がある. R-treesではノードの再構成は分割の時のみされる. この再構成はよい手段とは言えない. R*-treeではノードの再構成を改良するために強制再挿入(forced reinsertion)と呼ばれる手法を導入している. もしノードがオーバフローしてもすぐには分割しない. p個のエントリが削除され木に再挿入される. Beckmann[3]によればpは1ページあたりエントリの最大値の30%程度が最適である. 同時に, R*-treeでは同じ高さにあるノードの重なりが小さくなるように独自の分割アルゴリズムを改良している. しかし, 強制再挿入と分割アルゴリズムの改良はノードの再構成問題を完全には解決していない. なぜなら改良されたアルゴリズムは特定の状況下でしか効力を発揮しないからである. 依然としてオブジェクトをR*-treeの葉ノードとしてうまくクラスタリングさせる最適な手法がない.
2. ノードのエントリの最小値がクラスタリングの効果を低下させる.
   エントリの最小値は比較的良い空間利用率を保証するものである. しかし, この制限はR-treeの葉ノード

平成13年6月14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修士課程
*** 知能システム学部門

のクラスタリングを施した効果を低下させるものである．なぜなら，実際のアプリケーションではクラスタにごく少数のオブジェクトしか保持していないものもあり，そのグループが他のクラスタから離れていることがある．もし，クラスタに含まれるオブジェクトの数がノードのエントリ数の最小値よりも小さければこのクラスタは他の離れたグループと併合し一つの葉ノードを作らなければならない．この場合，葉ノードのMBRは巨大な大きさになり，葉ノード間の重なりも大きくなる．

3. オブジェクトの再挿入では必ずしも効率的なクラスタリングを行えるわけではない．

Fig.1ではこのノードで強制再挿入するつもりのオブジェクトが円の外側に表示されている．これらの数はこのノードに含まれるオブジェクト数の30%である．明らかにこの再挿入では効率的なクラスタリングができない．

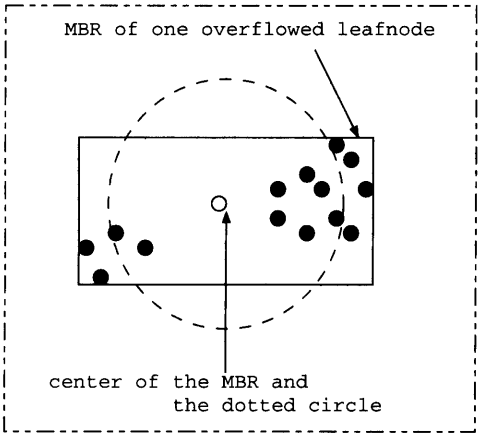

Fig.1 Choosing of reinserted objects in overflowed leaf node.

本論文では，前もってオブジェクトをクラスタリングする手法を取り入れている．クラスタリング手法として自己組織化マップSOM(Self-Organizing Maps)を用いた．ここで提案するSOMとR*-treeを組み合わせた手法は葉ノード間の重なりを減らし，オブジェクトをうまくクラスタリングすることができ，上記の欠点を克服できるものである．

ここで提案するSOM-based R*-treeの特徴には以下のようなものがある．

1. クラスタリング手法が非常に効果的で葉ノード間の重なりを大幅に減少させ，オブジェクトもうまくクラスタリングすることができる．
2. 葉ノードのサイズに制限がないためメモリを浪費しなくてすむ．「葉ノード」のサイズは実際のクラスタのサイズによって決定される．それゆえ，葉ノードの再挿入と分割が必要ない．

我々の実験結果によれば，この手法を用いると葉ノード間の重なりは大きく減少し，オブジェクトもうまくクラスタリングすることがてき，オリジナルのR*-treeと比較して検索時間が短縮される．特に，オブジェクトの距離計算が大きく減少する．この計算はマルチメディアデータベースの検索において最も時間がかかるものである．

SOMとR*-treeをもとにしたSOM-based R*-treeはすでに提案されている[4]．しかし，このSOM-based R*-treeは厳密なものではない．この問題については後述する．

本論文では全く新しいSOM-based R*-treeについて提案する．この手法は前述した欠点をすべて克服し，すべてのR*-treeアルゴリズムで利用可能なものである．

2. 関 連 研 究

関連研究としてクラスタリング技術，R*-tree，既存のSOM-based R*-treeについて紹介する．

2.1 クラスタリング

ニューラルネットワークでのクラスタリング手法には教師ありと教師なしの2つがある[5]．SOMは教師なしニューラルネットワークであり，2層構造からなる．第1層はN次元の入力層であり，第2層は複数のノードが2次元に配列されているマップ層である．マップ層では入力に合わせたN次元の重みベクトルを持ち，学習過程でこのベクトルが更新される．

SOMはデータを可視化したり高次元のデータを解釈するために使用されてきた[6),7),8),9),10),11)]．本研究でSOMを使用した理由は以下である．

1. データの知識が必要ない
2. 学習アルゴリズムが単純
3. 教師信号が不必要で自己学習を行う
4. SOMマップは類似したものを近くに配置する特徴を持つ

SOMマップは類似したものを近くに配置する位相特徴マップ(topological feature map)を形成する．類似したオブジェクトを検索するには質問点に最も近いノードである勝者ノード(winner node)の周りを探索すればよい．

SOMでは通常データベースの多次元特徴ベクトルを2次元平面にマッピングする．このマッピングではそれぞれのノードを代表する代表ベクトル1点に複数の特徴ベクトルを割り当て，クラスタリングされる．

Fig.2はSOMマップを表している．2次元平面は入力ベクトルが配置されマップ層を構成している．

マップ層のそれぞれのノードはコードブックベクトルと最整合リスト(BML:best matching list)があり，最整合リストにはそのコードブックベクトルに最も近いベクト

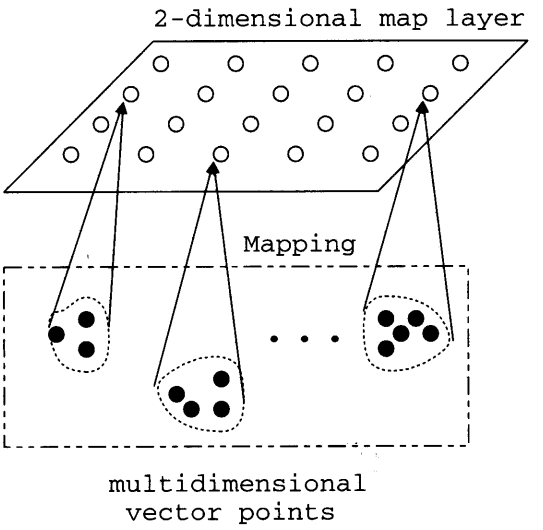

Fig.2 SOM map.

ルが入っている. マップ層のノードにはオブジェクトが1つも入っていない空きノードであることがある.

2.2 R-trees

R-treeファミリー, 特にその中でも最も有名なR*-treeは, よく使われる多次元インデックスの階層構造である.

R-treeはB-treeの多次元への拡張として提案された. R-treeはB-treeとQuadtree のよい特徴を組み合わせたものである. B-treeのように平衡木であり, 葉はすべて同じ高さになる.

Beckmannはデータ分散の違いによるR-treeの構造を研究し, R-treeの弱点を発見してその派生としてR*-treeを完成させた. そして, 彼は検索性能のためには挿入仮定が非常に重要であるというRoussopoulosとLeifkerの意見を立証した[12]. こうしてR*-treeに強制再挿入を取り入れた.

マルチメディアでのR*-treeはd次元のMBR(minimum bounding rectangle)を持った木構造である. MBRはオブジェクトの最小境界を表す矩形である. R-treeのインデックスノードにはそれぞれエントリの配列を保持している. エントリにはポインターとMBRが格納されている. 葉ノードにもエントリの配列があり, エントリにはオブジェクトとオブジェクト識別子が格納されている. **Fig.3**はR-treeの一例である.

2.3 Packed R*-tree

R-treeのパックアルゴリズムはRoussopoulos, Kamel と Faloutsos そしてScott T.Leutenegger によって提案された. この3つのアルゴリズムの共通点は以下のようなものである.

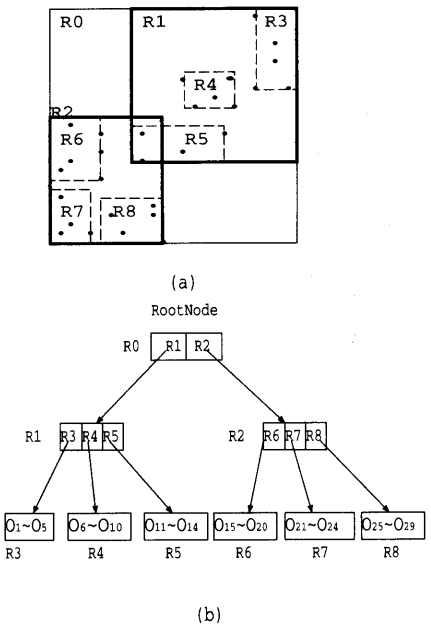

Fig.3 Example of R-tree (a) Whole data space (b) R-tree structure.

1. オブジェクト(または矩形)をb個のオブジェクトに分け, これらをグループ化して並べ替えるためにデータファイルの前処理を行う. ここでbはそれぞれの葉ノードに含まれるオブジェクト数である. 最後のグループはbよりも少ないかも知れない.
2. オブジェクト(または矩形)のグループすべてをページに読み込みそれぞれの葉ノードのMBRとページ数を一時ファイルに出力する. ページ数は上のレベルのノードに対するポインタに利用される.
3. この処理を再帰的に繰り返しながら上に組み上げていき, 根ノードを構築するまで続ける.

この3つのアルゴリズムの違いはそれぞれレベルでのオブジェクト(あるいは矩形)を並べる方法だけである. これらの手法の狙いはR*-treeの同レベルでのMBR の重なりを減らすことにある. しかし, これらのアルゴリズムではそれぞれの葉ノードにはすべて同数のオブジェクトが入っている. 当然これでは実際のオブジェクト分布を反映することができず, 良いクラスタリングもできない.

本論文での我々のアプローチはクラスタリング技術を用いてデータファイルを前処理することにある. オブジェクトの実際のクラスタがR*-treeの葉ノードとして利用される. そして, R*-treeを組む際に時間のかかるパックアルゴリズムを必要としない. 我々のアプローチを3.章で述べる.

2.4 既存のSOM-based R*-tree

前述した通り, SOM-based R*-treeは4)で提案された. この基本的な考え方は以下のようなものである.

- Step 1: SOMを用いて特徴ベクトルを事前にクラスタリングしておく
- Step 2: 空きノードは削除する
- Step 3:マップ層のコードブックベクトルをオブジェクトとしてR*-treeを構築する

4)ではSOM-based R*-treeとR*-treeのk-NN検索実験を行い, 性能比較をしている. しかし, このSOM-based R*-treeは厳密ではない.

この手法ではまずオブジェクトがクラスタリングされ, そのクラスタがコードブックベクトル(CBV:codebook vector)を持っている点にマッピングされる. このCBVを用いてR*-treeを構築する. k近傍探索(k-NN search: k-nearest neighbor search)を行う場合は, このSOM-based R*-treeに対して施され, k近傍のCBVが取り出される. もちろんこれが最終的な結果ではない. 取り出されたものはクラスタでしかなく, 実際に欲しい結果はk個のオブジェクトなのだ. もしかすると検索結果に含まれていないクラスタの中に欲しい検索結果があるのかも知れない. 常に質問点に近いグループに質問点に近いオブジェクトがマッピングされているとは限らない. それゆえ, このSOM-based R*-treeの検索結果には欲しい結果が含まれていない可能性がある. 我々の実験結果によれば検索結果の一致度は約70%から85%程度である. このような結果は一部の応用に対して十分であるかも知れないが, 厳密ではない.

もう一つの欠点としてこのSOM-based R*-treeでは階段的近傍探索アルゴリズム(INN search algorithm : Incremental Nearest Neighbor search algorithm)を使用できないことがある. このアルゴリズムはどの程度近傍数が欲しいのかがまだわかっていないときに用いるものであり, 近傍数kを階段的に増加させることができる.

上記の欠点をすべて克服したSOM-based R*-treeを本論文では提案する.

3. 新しいSOM-based R*-tree

3.1 新しいSOM-based R*-treeとは

R*-treeでは葉ノードはいくつかの特徴ベクトルから構成される. 空間利用率の保証のために, ノードのエントリ数は最小値mと最大値Mの間に制限される. しかし, 本論文で提案する新しいSOM-based R*-treeはSOMによってクラスタリングされたものをそのまま「葉ノード」として利用する. 葉ノードのエントリ数の制限が省略されているにも関わらず利用空間は保証される. このため, バッファにはクラスタリングされたものすべてが入ることになる.

Fig.4(a)はSOMによってクラスタリングされたものであり, (b)は対応した新SOM-based R*-treeを表している. この図ではmは対応するクラスタ中のオブジェクト数,

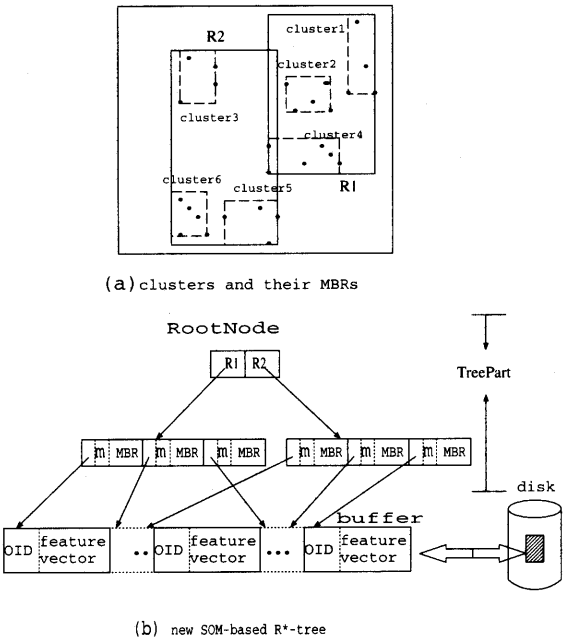

Fig.4　New SOM-based R*-tree.

MBRはそのクラスタのMBRを表している. バッファ中のクラスタはSOM-based R*-treeの「葉ノード」と見なされる. これらの葉ノードの集合は実質的なオブジェクトの分布を良く反映しており, 葉ノード間の重なりは大きく減少する. したがって, k-NN探索時のプルーニングはより有効になる.

3.2 構築

SOM-based R*-treeの構築手順は次のようになる.

- **Step1:** SOMを用いてすべてのオブジェクトをクラスタリングする
- **Step2:** クラスタのMBRを計算する
- **Step3:** Step2で得られたMBRを用いて「オブジェクト」としてR*-treeを構築する
- **Step4:** すべてのクラスタを含んだバッファを構築する
- **Step5:** Step3で得られたR*-treeとStep4で得られたバッファを結合する

この手法ではデータベースが変化すると, クラスタリングなどの前処理を再度行い, SOM-based R*-treeを再構築しなければならない. この問題の解決は,我々の次の研究課題になっている.

3.3 検索

一般的にすべてのR*-treeの検索アルゴリズムは少しの変更で新SOM-based R*-treeに適用できる. ここで, N.Roussopoulos[13]が提案したk-NN探索アルゴリズムを例として取り上げる. 新SOM-based R*-tree用に改良し

たk-NN探索アルゴリズムを **Fig.5**に表示する.

Fig.5では, qが質問オブジェクトで, インデックスツリーが用いている全体インデックス, NearListが候補リストである. NearListはアルゴリズムが終ると最終的な探索結果になる. 葉ノードは, **Fig.4**でのバッファ以外のツリーの葉ノードを示す. サブアルゴリズムk-NT()は, 再帰的に引数とするNodeのサブツリーを辿りながら, 候補リストを更新する. メインアルゴリズムk-NN()は, サブアルゴリズムk-NT()を呼び出し全体のインデックスを辿る.**Fig.5**のアルゴリズムとN. Roussopoulos [13]が提案したk-NNアルゴリズムの違いは, 行2 ～ 行6 部分のみである. つまり, 葉ノードに到着すると, バッファの対応したクラスタを訪問しなければならない.

```
Main procedure:: k-NN(q, k, IndexTree)
/* k-nearest neighbor search in IndexTree */
(1) NearList=new List(k)
(2) Distance of each member in NearList ← ∞
    /* initialize NearList */
(3) k-NT(q,k,NearList,RootNode)
(4) report all objects in NearList
END
Subprocedure:: k-NT(q,k,NearList,Node)
1 If Node is a leaf node
2   For each entry in Node Do
3     If Dist(q,MBR)<NearList.MaxDist
4       For each object of the corresponding cluster
5         If Dist(q,object)<NearList.MaxDist
6             insert(NearList, Object)
                  /* update NearList */
7 if Node is not a leaf node
8     ActiveBranchList ← child_nodes in Node
9     Sort_by_distance(ActiveBranchList)
10    last=pruneActiveBranchList;
//last refers to the number of child_nodes
//left after pruned
11  For i:=1 to last
//for each child_node left in ActiveBranchList
12      if Dist(q, child_node_i)≤NearList.MaxDist
13          k-NT(q,k,NearList,child_node_i)
14      else
15          exit For-loop
END
```

Fig.5 Adopted k-NN search algorithm on new SOM-based R*-tree.

4. 実 験 結 果

新SOM-based R*-treeの構築時間とk-NN探索アルゴリズムの動作を検証するためにカラー画像データベースを用いた.同じデータベースでのオリジナルR*-treeも構築し, その探索性能を考察した. また, 比較はR*-treeと新SOM-based R*-treeで行っている.

画像にはH^2soft [14]会社と Stanford大学 [15]のものを用いた. これらには風景, 動物, 建物, 人, 植物など様々な画像が含まれている. 画像のサイズは128 × 128としている. 特徴抽出手法としてウェーブレット変換の一種であるHaarウェーブレット[16]を用いた. Haarウェーブレットは計算時間が早く, 実用的であるためである[17]. 6レベルの2次元ウェーブレット変換を用いることにより画像の特徴ベクトルは12 次元まで落とされる.

質問画像はデータベース中からランダムに100枚選び, その他の画像でインデックスの木を構築した.

検索時間とは別にオブジェクト間の距離計算の回数も計測した. これはマルチメディア検索において最も時間のかかるものであると見なされている. これらの実験はCOMPAQ DESKPRO i386 (OS:FreeBSD 3.4-STABLE) Memory 128MB, 画像数は40000で行った.

1. インデックス構築の比較

Table 1 Comparison of building index.

items	R*-tree	SOM-based R*-tree
timecost	52.5536(sec.)	17.5602(sec.)
height	4	4
index size	13.5MB	8.111MB

Table 1ではインデックス構築に必要なデータをメモリ中に読み込んでからの時間を計測している. つまり, I/Oは含まれていない. 読み込まなくてはならないデータはR*-treeも新SOM-based R*-treeも同じなのでI/Oの時間も当然同じである. 新SOM-based R*-treeのインデックスサイズは必要なバッファサイズも含んでいる.

この結果によると, R*-treeと比較して新提案SOM-based R*-treeはインデックスの構築が速くなった.ファイルのサイズも小さくなった.

2. 検索性能比較

有名なk-NN検索を例として取り上げた. kの値を変化させ, R*-treeと新SOM-based R*-treeの検索性能を比較実験した.

Fig.6ではk-NN探索性能比較を表している.

この結果によると, R*-treeと比較した場合, 新提案SOM-based R*-treeは探索時間が約半分になり, オブジェクト間の距離計算の数が1/4~1/10となって大幅に改良で

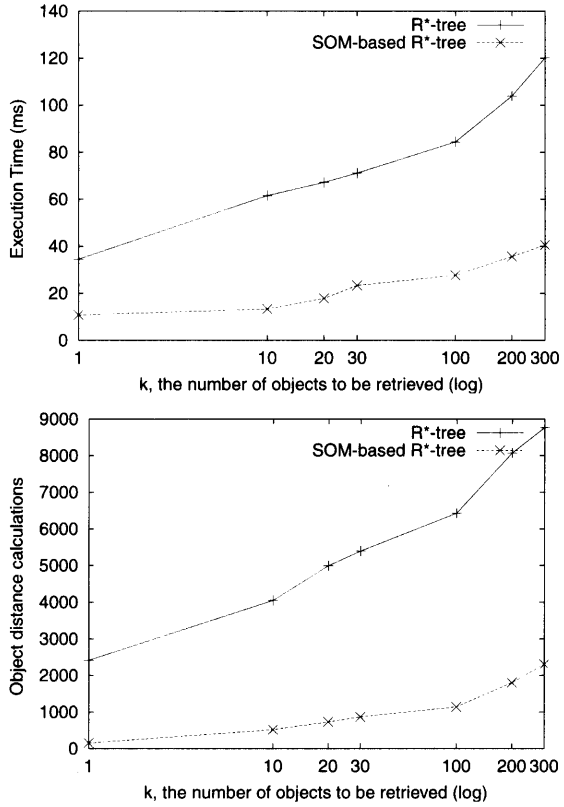

Fig.6 comparison on execution time and object distance caculations.

きた.

5. むすび

本論文ではSOMとR*-treeを組み合わせて葉ノードの重なりを減らした. 葉ノードでのオブジェクトの集合は実際の分布を良く反映している. 検索性能を大幅に改良させた. この手法ではデータベースが変化すると, クラスタリングなどの前処理を再度行い, SOM-based R*-treeを再構築しなければならない. ここで提案したSOM-based R*-treeは本実験でR*-treeよりも非常によい検索性能があることがわかった. 新提案SOM-based R*-treeの構造, 構築, そして検索アルゴリズムについて詳細に述べた.

参考文献

1) *J.C. Russ "The Image Processing Handbook"*. CRC Press, Boca Raton, 1995.
2) A. Guttman. "R-trees: A Dynamic Index Structure for Spatial Searching". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 47–57, June 1984.
3) N. Beckmann, H.P. Kriegel, R. Schneider, B. Seeger. "The *R**-tree: An Efficient and Robust Access Method for Points and Rectangles.". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 322–331, May 1990.
4) K. Oh, Y. Feng, K. Kaneko, A. Makinouchi. "SOM-Based R*-Tree for Similarity Retrieval". In *Proceedings of DASFAA2001*, pages 182–190, Hongkong, China, May 2001.
5) R.O. Duda, P.E. Hart. *" Pattern Classification and Scene Analysis"*. John Wiley & Sons, 1973.
6) A. Rauber. "LabelSOM: On the Labeling of Self-Organizing Maps". In *Proceedings of IJCNN'99*, Washington DC, July 1999.
7) T. Kohonen. "Self-Organization of Very Large Document Collections: State of the Art". In *Proceedings of ICNN98*, volume 1, pages 65–74, London, UK, 1998.
8) T. Kohonen. "Exploration of Very Large Databases by Self-organizing Maps". In *Proceedings of ICNN'97*, Piscataway, NJ, 1997. IEEE Service Center.
9) G. Deboeck, T. Kohonen. *" Visual Explorations in Finance with Self-Organizing Maps"*. Springer-Verlag, London, 1998.
10) S. Kaski. "Dimensionality Reduction by Random Mapping: Fast Similarity Computation for Clustering". In *Proceedings of ICNN'98*, pages 413–418, Piscataway, N-J, 1998.
11) S. Kaski. "Fast Winner Search for SOM-Based Monitoring and Retrieval of High-Dimensional Data". In *Proceedings of ICANN'99*, Edinburge, UK, September 1999.
12) N. Roussopoulos, D. Leifker. "Direct Spatial Search on Pictorial Databases Using Packed R-trees". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 17–31, 1985.
13) N. Roussopoulos, S. Kelley, F. Vincent. "Nearest Neighbor Queries". In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 71–79, San Jose, CA, June 1995.
14) *H^2soft, http://www.h2soft.co.jp.*
15) *Standford University, http://www-DB.Standford.EDU/IMAGE.*
16) S.G. Mallat. "Multifrequency Channel Decompositions of Images and Wavelet Models". In *IEEE. Transactions on Acoustic Speech and Signal Process*, volume 37, Number 12, pages 2091–2110, 1998.
17) C.E. Jacobs, A. Finkelstein, D.H. Salesin. "Fast Multiresolution Image Querying". In *Proceedings of SIGGRAPH95*, pages 6–11, Los Angeles, California, 1995.