DEIM Forum 2009 D8-5

オブジェクトごとの近傍情報を用いたk-NN探索手法

山本　俊介†　　遠山　元道††

† †† 慶應義塾大学理工学部情報工学科　〒 223–8522 神奈川県横浜市港北区日吉 3–14–1
E-mail: †shunsuke@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし　k-NN 探索は，空間上に指定された質問点に近接するオブジェクトを空間データベースの中から k 個求める探索である．このような探索は空間データ構造を用いることで効率よく実行できる．一方，そのようにして求めた k-NN 探索の結果をあらかじめ空間データベースの各オブジェクトに近傍情報として保有させておくことにより，データベースへの問い合わせ結果の周辺のオブジェクトに容易にアクセスすることが可能となる．しかしこの場合は，オブジェクトがデータベースに追加，削除，更新等されるたびに各オブジェクトの近傍情報を更新する必要が生じる．本稿では，各オブジェクトに近傍情報を保有させ，さらにデータベースの更新が起きた場合にその情報を効率よく更新する手法を提案する．

キーワード　データ構造，インデックス

k-NN Search Method using Nearby Information on each Object

Shunsuke YAMAMOTO† and Motomichi TOYAMA††

† ††Department of Information and Computer Science, Faculty of Science and Technology,
Keio University
Hiyoshi3–14–1, Kouhoku-ku, Yokohama-shi, Kanagawa, 223–8522 Japan
E-mail: †shunsuke@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

Abstract　The k-NN search is a search for the k objects that are close to the question point specified on the space from the spatial database. Such a search can be efficiently executed by using the spatial data structure. On the other hand, easily accessing the objects around the result of a query becomes possible with having each object of the spatial database have the result of the k-NN search beforehand as nearby information. However, the necessity for updating the nearby information whenever the object is added, deleted, or updated to the database is caused in this case. In this paper, we have each object have the nearby information, and in addition, we propose the method for efficiently updating the nearby information when the update of the database occurs.

Key words　Data Structure, Index

1. はじめに

k-NN(k Nearest Neighbor) 探索とは，空間データベースの中から指定されたオブジェクトに近接する空間オブジェクトを近い順に k 個求める演算である．この探索は GIS(Geographic InfomationSystem:地理情報システム) や CAD の分野における基本的かつ重要な演算のひとつであるため，これまでに多くの研究が行われてきた．

k-NN 探索を高速に行うための手段のひとつとして，空間データ構造を用いたインデックス作成が挙げられる．空間データ構造とは，空間データベースを互いに近接性のあるデータから成るグループ (ページと呼ぶ) に分割し，データ検索を効率化するためのデータ構造である．現在広く用いられている空間データ構造として，R-tree [1] やその改良型である R*-tree [2] などが挙げられる．また SR-tree [3] は，R-tree において矩形領域であらわされていたエントリーを矩形領域と球形領域の重なり合った部分で定義することにより索引構造内のオーバーラップを減少させ，さらに良い検索効率を実現している．

これらのデータ構造を対象として，Roussopoulos らは深さ優先探索に基づいた k-NN 探索アルゴリズムを提案している [4]. これは，木の各レベルにおいて検索点に最も近接するノードを選択しつつ木を下に向かって辿り近接オブジェクトを探す手法である．また，Hjaltason [6] や Berchtold [5] らは，優先順位付きキューにノードを挿入することによって，常に最良な順序で木の探索を行うことが可能な方式を提案している．このような手法は，先に述べた深さ優先 (depth-first) 探索に対して，最良

優先 (best-first) 探索と呼ばれる．

一方，ユーザがデータベースへの問い合わせによってある結果（オブジェクト）が得られたとき，ユーザは当該オブジェクトだけでなく，その周辺のオブジェクトの情報も求めることが多いと考えられる．その場合，結果オブジェクトからその周辺のオブジェクトにも容易にアクセスできると有用である．そのためには各オブジェクトがあらかじめ自身の近傍情報を保有しておけばよい．本稿ではその近傍情報として各オブジェクトごとの k-NN 探索の結果を持たせる．この情報をオブジェクトごとの k-NN 情報と呼ぶこととする．しかしこの場合，k-NN 情報の作成にかかるコストや，オブジェクトの追加，削除等に伴う k-NN 情報の更新方法についても考えなければならない．本稿では，実際にオブジェクトに k-NN 情報を保有させ，さらにデータベースの更新が起きた場合にその情報を効率よく更新する手法を提案する．

以下，本稿の構成を示す．まず 2 章で各オブジェクトに保有させる k-NN 情報について述べる．次に 3 章で k-NN 情報の更新について述べる．そして 4 章で k-NN 情報を更新すべきオブジェクトを探索する手法について述べ，続く 5 章で実験とその結果を述べ，6 章で結論およびまとめを述べる．

2. k-NN 情報

2.1 k-NN 情報

ユーザがデータベースへの問い合わせによってある結果（オブジェクト）が得られたとき，ユーザは当該オブジェクトだけでなく，その周辺のオブジェクトの情報も求めることが多いと考えられる．その場合，結果オブジェクトからその周辺のオブジェクトにも容易にアクセスできると有用である．

近傍情報を得るためには，当該結果オブジェクトを質問点として k-NN 探索を行えばよいが，さらに高速に情報を得るには各オブジェクトがあらかじめ自身の近傍情報を保有しておけばよい．そこで近傍情報へのアクセスを容易にするために，各オブジェクトに以下の 2 つの情報を保有させることを考える．この情報を各オブジェクトごとの「k-NN 情報」と呼ぶ．

- k-NN オブジェクト
- k-NN オブジェクトまでの距離

ここで k の値は任意である．k-NN 情報は全オブジェクトにおいて k-NN 探索を実行することで取得する．当然，この k-NN 探索を高速に行うために空間データに対して索引（$SR-tree$）を作成しておく必要がある．

2.2 k-NN 情報の例

各オブジェクトに k-NN 情報を持たせる例として図 1 のような二次元オブジェクト集合を考える．仮に k=5 とすると自身に近いオブジェクト 5 つとそこまでの距離を全オブジェクトが保有することになる．例として図 1 中のオブジェクト A の k-NN 情報を表 1 に，オブジェクト B の k-NN 情報を表 2 に示す．三次元以上の多次元においても，同様な表が作られる..

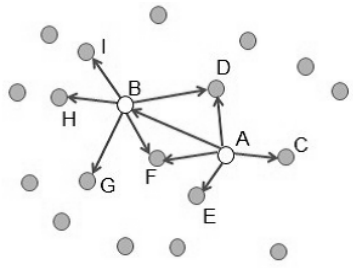

図 1　二次元オブジェクト集合

表 1　オブジェクト A の k-NN 情報 (k=5)

	1-NN	2-NN	3-NN	4-NN	5-NN
id	E	C	D	F	B
distance	0.3	0.4	0.5	0.7	1.2

表 2　オブジェクト B の k-NN 情報 (k=5)

	1-NN	2-NN	3-NN	4-NN	5-NN
id	F	I	H	D	G
distance	0.3	0.4	0.5	0.9	1.0

3. k-NN 情報の更新

3.1 k-NN 情報の更新の必要性

前章で述べたような方法で求めた k-NN 情報であるが，対象となる多次元データに変化が起きると，その更新の必要性が生じる．多次元データの変化とは，具体的に言うとオブジェクトの追加または削除である．以下，それらの変化が起こった場合に k-NN 情報をどのように更新するかについて述べていく．

3.2 オブジェクトの追加

k-NN 情報を持つオブジェクト集合に新たにオブジェクトが追加された状況を考える．当然追加オブジェクトは k-NN 探索を行い k-NN 情報を取得する必要があるが，それだけでなく既存オブジェクトの中にも自身の k-NN 情報の更新をしなければならないものがある．そのようなオブジェクトを「更新オブジェクト」と呼ぶこととする．更新オブジェクトを知るのは次節に述べる削除の場合と比較して容易ではない．

更新オブジェクトを知るために，まず「近傍円」という考えを導入する．近傍円とは中心および半径を以下のように定義した円である．

- 近傍円

・中心　各オブジェクト

・半径　k-NN 情報のうち 1 番遠いオブジェクトまでの距離（以下「$k-dis$」と呼ぶ）

既存オブジェクトのうち，k-NN 情報を更新すべきか否かはこの近傍円を利用して決定される．すなわち，以下の 2 パターンが考えられる．

（1） 追加オブジェクトが近傍円の内部の場合

k-NN 情報の更新が必要である．さらにその更新によって近傍円の半径は小さくなる．

（2） 追加オブジェクトが近傍円の外部の場合

k-NN 情報の更新は不要である．

図 2 はオブジェクト A とその近傍円を表している．近傍円の半径は A の k-NN 情報において最も遠いオブジェクトまでの距離 $(k-dis)$ となっている．ここにオブジェクト B または C を追加した状況を考える．B を追加した場合は A の近傍情報を更新する必要はないが，C を追加した場合は更新しなければならない．B は近傍円の外部に存在するのに対し，C は A の近傍円内に存在するためである．つまり，追加オブジェクトと近傍円との包含判定を行うことで更新すべきか否かが定まるのである．

C を追加した後の A とその近傍円の様子を図 3 に示す．この場合は A の k-NN 情報が更新され，それに伴い $k-dis$ も更新されるため，結果として近傍円の半径は小さくなっている．

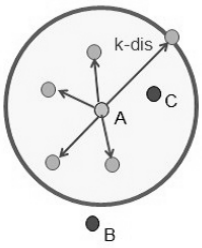

図 2　オブジェクトの追加

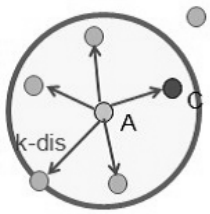

図 3　オブジェクト C 追加後

3.3　オブジェクトの削除

続いて，k-NN 情報を持つオブジェクト集合からオブジェクトが削除された状況を考える．この場合は追加の際よりも比較的容易に k-NN 情報の更新が可能である．具体的には以下の手順で行われる．

（1） 削除オブジェクトを k-NN 情報に含むオブジェクト（更新オブジェクト）を検索する．

（2） 更新オブジェクトがそれぞれ新たに k-NN 探索を行い自身の k-NN 情報を更新する．

オブジェクトの追加の場合は更新オブジェクトの探索に近傍円という概念を用い，追加オブジェクトと近傍円との包含判定をしなければならなかった．しかし削除の場合は更新オブジェクト探索は容易である．なぜなら既存オブジェクトの k-NN 情報に削除オブジェクトが含まれているかどうかのみを調べればよいためである．

3.4　更新オブジェクト数

オブジェクトを追加した場合に更新が必要なオブジェクトの数はいかほどになるのか，実験を行い調べた．k-NN 情報を持つ多次元一様分布の点集合に対し新たに点を追加するという動作を 100 回行い，更新オブジェクト数の平均を計測した．点の数は 10000 個とし，k の値は 1,5,10 の 3 パターンで行った．次元数は 2 から 100 まで変化させた．

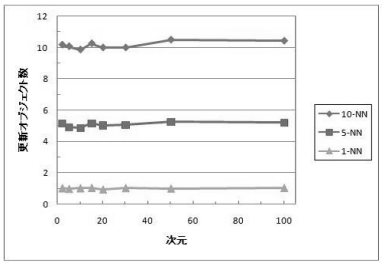

図 4　更新オブジェクト数

その結果を示した図 4 を見ると，次元に関係なく更新オブジェクト数は k の値にほぼ一致していることが分かる．さらに，点の数を 10000 個から変化させてもこれと同様な結果が得られた．そのため更新オブジェクト数は平均的には k の値に一致し，k が増えるほどその数も増えるといえる．

4.　更新オブジェクト探索

4.1　全件走査法

前章で述べたように，多次元データにオブジェクトを追加した場合，k-NN 情報の更新を行うためには更新オブジェクトの探索が必須となる．更新オブジェクトを探索するための手法としてまず考えられるのは，以下に述べる単純な全件走査法である．

- 全件走査法

（1） 追加オブジェクトと各オブジェクトとの距離 (dis) を計算する．

（2） dis と各オブジェクトの $k-dis$ を比較する．

（3） $dis < k-dis$ ならば当該オブジェクトは更新オブジェクトである．

このように全オブジェクトとの距離を測ることにより更新オブジェクトを知ることができる．しかし更新すべきオブジェクトの数は全オブジェクト中のほんの一部にすぎない．図 4 で表されているように，更新すべきオブジェクトの数は k の値にほぼ一致するからである．そのような数少ない更新オブジェクトを探索するために，全オブジェクトとの距離を計算するのは非効率である．そこで更新オブジェクトの探索範囲を絞り込むことを考える．

4.2　空間索引を用いた更新オブジェクト探索

4.2.1　空間索引の作成

更新オブジェクトの探索範囲を絞り込むために，前章で述べた近傍円の概念を用いる．更新オブジェクトを知るためには，追加オブジェクトと近傍円との包含判定をすればよいのであった．そこで各オブジェクトの近傍円を空間オブジェクトと捉え，それらに対し空間索引を作成することとする．このような索引を作成しておけば，追加オブジェクトが近傍円に含まれるかどうかの包含判定を高速で行える．

空間索引を作成する過程を図 5 に示す．図 5 ではまず二次元オブジェクト集合に対し例として 1-NN 情報を持たせ，その

近傍円を生成している．続いて各近傍円に対し空間索引として $R-tree$ を作成する．その際，各近傍円を包囲する最小の矩形が MBR となる．

図 5 では二次元の場合を想定しているが，三次元以上の多次元でも同様な方法で空間索引の作成が可能である．また，k の値を変化させるとどうなるかというと，k が大きくなるほどより大きな近傍円，及びそれを包囲する MBR が生成されることになる．近傍円の半径が大きくなると，近傍円間の重なりが大きくなる．すると追加オブジェクトを包含する近傍円の数が増え，その分更新オブジェクト数も増えるのである．

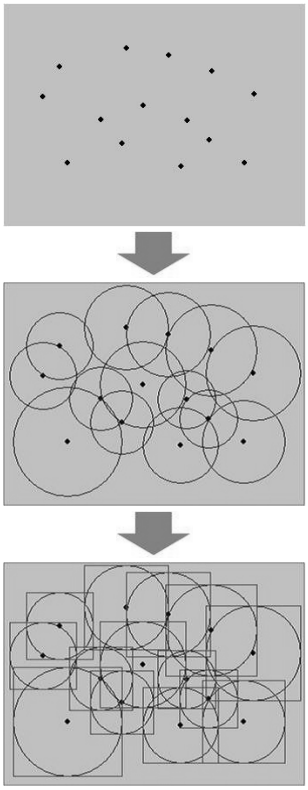

図 5　R-tree の作成

4.2.2　R-tree を用いた探索

このようにして作成された $R-tree$ を用いて更新オブジェクトを探索する手法を以下に示す．

（1） 追加オブジェクトがどの MBR に包含されるか $R-tree$ を用いて探索する．

（2） 包含判定の結果，該当したオブジェクトは更新オブジェクトの候補となる（「更新候補オブジェクト」と呼ぶ）．

（3） 該当した MBR 内で追加オブジェクトが真に近傍円に包含されるかどうかを，更新候補オブジェクトとの実際の距離を測定することにより判別する．

（4） 追加オブジェクトが近傍円にも包含されるならば当該オブジェクトは更新オブジェクトである．

図 6 は k-NN 情報を持つ二次元オブジェクト集合に新たにオブジェクト C が追加された様子を表している．ここでは $k=1$ としている．図中には既存オブジェクト A および B の 1-NN の近傍円と MBR を示した．なお，近傍円にはあらかじめ空間索引として $R-tree$ が作成してある．

ここで，追加オブジェクト C は既存オブジェクト A および B の MBR に包含されるため，A，B は更新候補オブジェクトとなる．続いて C と A，B との実際の距離を測定する．すると C は B の近傍円には包含されないことがわかる．そのため B は更新の対象とならない．それに対して C は A の近傍円には包含されるため，A は更新オブジェクトとなる．

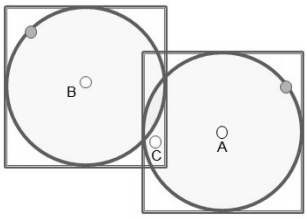

図 6　更新オブジェクト探索

このようにして探索された更新オブジェクトはそれぞれ新たに k-NN 探索を行い，自身の k-NN 情報を更新することとなる．オブジェクト C 追加後の近傍円および MBR の様子を図 7 に示す．

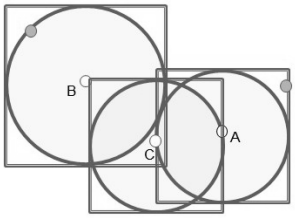

図 7　オブジェクト C 追加後

5.　実験と評価

5.1　一様分布データ

本節では多次元一様分布点データを用意して実際に k-NN 情報を持たせる実験を行った．点データの概要は以下のとおりである．

- 点の数 n は 10000 点と 50000 点
- 次元数は 2 から 100 まで
- k の値は 1, 5, 10 の 3 パターン

これら点オブジェクトに k-NN 情報を持たせるために，索引として $SR-tree$ を作成した．

5.1.1　k-NN 情報作成時間

$SR-tree$ を用いて全オブジェクトが k-NN 探索を行い，各オブジェクトに k-NN 情報を持たせた．10000 点，50000 点のそれぞれの場合において，全オブジェクトが 10-NN 情報を保有するまでに要した時間を図 8 に示す．

図 8 を見ると，オブジェクトの数によらず k-NN 情報作成時間は次元数に比例して増していくことがうかがえる．k の値を変化させても作成時間にほとんど変化がなく，ほぼ同様なグラフが得られた．

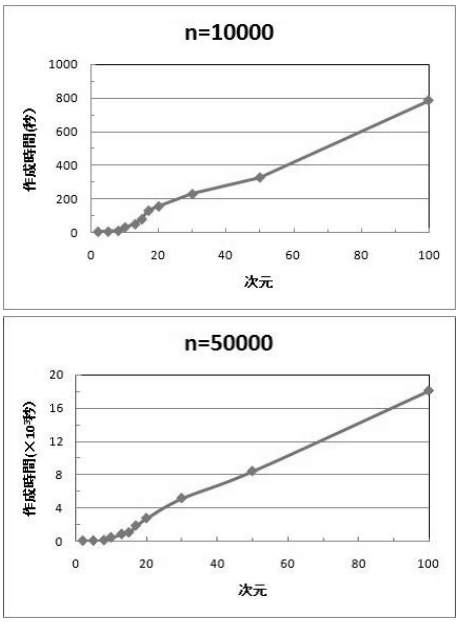

図 8　k-NN 情報作成時間

5.1.2　k-NN 探索時間

この一様分布集合中のオブジェクトに対し，以下の二つの方法で k-NN 探索を行った．

（1）　k-NN 情報を利用（提案手法）

（2）　$SR-tree$ を利用（従来手法）

それぞれの場合の k-NN 探索時間の結果を図 9 に示す．

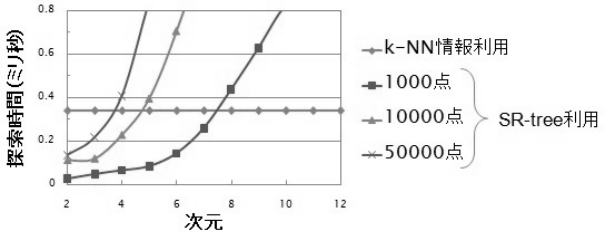

図 9　k-NN 探索時間

k-NN 情報利用の場合は，次元数やオブジェクト数に関係なく探索時間はほぼ一定となった．$SR-tree$ 利用の場合は高次元になると探索時間が飛躍的に増し，また，オブジェクト数が多いほど探索時間が増した．よって，提案手法は安定して高速な探索が可能であるが，低次元等の一部の条件下では従来手法の方が高速であるといえる．

5.1.3　オブジェクトの追加

この一様分布オブジェクト集合に点オブジェクトをランダムに追加し，更新オブジェクトの探索を行った．探索には前章で説明した以下の 2 つの手法を用いた．

- 手法 1

全件走査法 (追加オブジェクトと既存の全オブジェクトとの距離を計算する)

- 手法 2

あらかじめ近傍円に対し空間索引を作成しておくことで，探索範囲の絞り込みを行う手法

それぞれの手法でオブジェクトの追加を 100 回行った場合の，更新オブジェクト探索の平均所要時間を図 10 に示す．なお，手法 2 で用いる空間索引は R^*-tree とした．

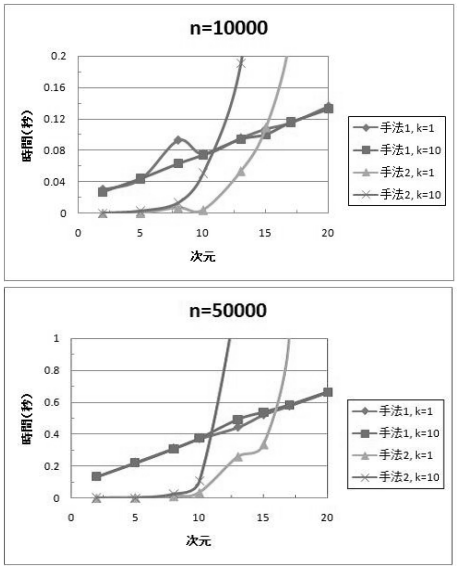

図 10　更新オブジェクト探索時間

図 10 から，10 次元程度までであれば手法 2 の方が優れているが，それ以上の高次元では急激に性能が劣化し，手法 2 の方が優れることがわかる．また，k の値が大きい方が探索に多くの時間を要している．さらに 10000 点と 50000 点とを比べると，後者の方が手法 2 の性能が手法 1 に劣るときの次元が高くなっている．これらの原因について，以下考察する．

5.1.4　高次元で性能が劣化する理由

一様分布においては高次元になるほどオブジェクト間の距離が増す．それに伴い R^*-tree において近傍円を包囲する MBR も巨大化していく．すると MBR と近傍円（球）との間で，体積の差が広がっていくことになる．つまり MBR の体積が近傍円（球）の体積に比べかけ離れて大きい値になってしまうのである．このような状態で追加オブジェクトと MBR との包含判定を行うと，多数の MBR が追加オブジェクトを包含すると判定してしまう．そのため更新オブジェクトの候補にはなるものの，実際には更新する必要がないオブジェクトの割合が増していくのである．

そのことを示すため，手法 2 によって絞り込める更新候補オブジェクト数の全既存オブジェクト数に対する割合を調べた (図 11)．

図 11 を見ると，図 10 と同様に，10 次元を超えたあたりから急激に候補オブジェクト数の割合が高くなっている．このことから，次元が増すほど更新候補オブジェクトの絞り込みは機能しなくなっていくことがうかがえる．高次元では結局ほぼすべてのオブジェクトとの距離を測っており，これでは全件走査法と変わらない．

ただ，10000 点と 50000 点とを比べると似たグラフではあるものの，後者の方が割合が高くなるときの次元が高い．これはオブジェクトの数が多いほどオブジェクト同士が密集するため

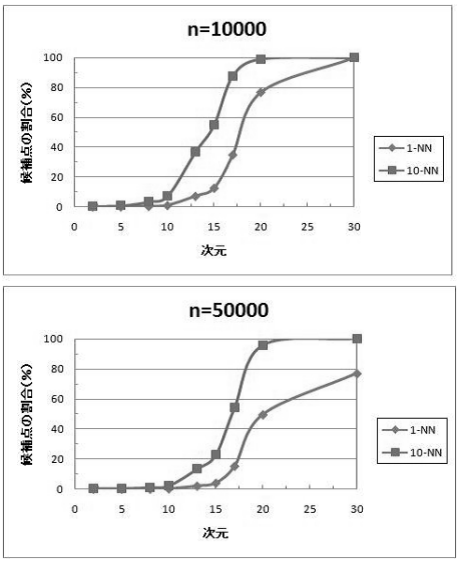

図 11　手法 2 で絞り込める更新候補オブジェクト数の割合

である．密集しているということはオブジェクト間の距離が小さいということである．すでに述べたように，手法 2 はオブジェクト間の距離が小さいほど機能する．このことは低次元ほど性能が良いことにも表れている．

以上のことをまとめると，手法 2 の性能を左右する要因は

（1） 次元数

（2） k の値

（3） オブジェクトが存在する密度

の 3 つであるといえる．

要因 1，2 についてはその値が小さければ小さいほど，要因 3 についてはその値が高ければ高いほど性能は良くなる．

5.2 実データ

続いて図 12 のような北米の文化的建造物を示す二次元点データ 204,493 件に対し，実際に k-NN 情報を保有させ，オブジェクトの更新を行った．

図 12　北米の文化的建造物のデータ

まず $SR-tree$ を用いた全オブジェクトの k-NN 情報作成時間は，$k=1$ では 2.9 秒，$k=10$ では 3.5 秒，$k=100$ では 9.5 秒となった．また，手法 2 を用いるための R^*-tree 作成時間は，$k=1$ では 509 秒，$k=10$ では 664 秒，$K=100$ では 538 秒となった．

そして 10-NN 情報を持たせた集合に点オブジェクトを 100 件追加した場合の更新オブジェクトの平均探索時間を調べたところ，手法 1 では 12.1 秒なのに対し手法 2 では 0.2 秒となった．手法 2 が手法 1 に比べ格段に高速である主たる理由は，対象が二次元データであることである．前節で示した通り，低次元であればあるほど手法 2 の性能は良くなるのである．このように近傍円の空間索引の作成にはコストを要するものの，手法 2 を用いることによって k-NN 情報の更新処理時間は大幅に改善されることが示された．

6. ま と め

空間データベース中の各オブジェクトに自身の k-NN 情報を保有させると，検索結果となるオブジェクトの周辺オブジェクトに容易にアクセスすることができるため有用である．しかしその場合はオブジェクトの追加や削除が起きると k-NN 情報の更新が必要となる．

本論文ではオブジェクト集合に実際に k-NN 情報を持たせた場合のコストを調べ，どのような場合にどのオブジェクトの k-NN 情報の更新が必要となるかを示した．更新すべきオブジェクトを効率的に探索する手法として，近傍円を空間オブジェクトとして捉える手法を提示した．この手法は低次元などの一定の条件下において，単純な探索よりも格段に高速に探索可能であるという結果が得られた．さらに現実の二次元データにも適用し，実際に探索が高速化されていることも示した．

今後の展望としては，k-NN 情報を持つことのさらなる意義の探求や，$R*-tree$ 以外の空間索引の作成による更新オブジェクト探索時間の測定などが考えられる．

文　献

[1] A. Guttman: “ R-trees: a dynamic index structure for spatial searching ”, *Proceedings ACM SIGMOD Conference on Management of Data*, pp. 47-57, 1984

[2] N. Beckmann, H. P. Kriegel, R. Shneider and B. Seeger: “ The R*-tree: an efficient and robust method for points and rectangles ”, *Proceedings ACM SIGMOD Conference on Management of Data*, pp. 322-331, 1990

[3] Norio Katayama, Shin ’ichi Satoh.: ”the SR-tree: An Index Structure for High-Dimensional Nearest Neighbor Queries”, *Proceedings ACM SIGMOD Conference on Management of Data*, pp.369-380, 1997

[4] Nick Roussopoulos, Stephen Kelley, and Frederic Vincent: “ Nearest Neighbor Queries ”, *Proceedings ACM SIGMOD Conference on Management of Data*, pp. 71-79, 1995

[5] Stefan Berchtold, Christian Bohm , Daniel A. Keim, and Hand-Peter Kriegel: “ A Cost Model For Nearest Neighbor Search in High-Dimentional Data Space ”, *Proceedings of PODS ’97*, pp. 78-86, 1997

[6] Gisli R. Hjaltason and Hanan Samet: “ Ranking in Spatial Databases ”, *Proceeding of the 4th Symposium onn Spatial Databases*, pp. 83-95, 1995