D-17 近傍連鎖点列の静的な索引における差分更新

佐藤 康裕† 伊藤 晶規†遠山 元道‡

†慶應義塾大学大学院 理工学研究科 開放環境科学専攻 ¶慶應義塾大学 理工学部 情報工学科

1 はじめに

近年のインターネット利用者とコンピュータの記憶容量の増大により、個人的なものやインターネットで流通する画像や音声などのマルチメディアデータが急速に増大してきた。それに伴い、蓄積されたマルチメディアデータを効率良く利用するための検索技術の必要性が増している。マルチメディアデータの検索にはデータの特徴量を抽出して多次元空間内に配置し、与えられた質問点からの距離が近い点たかだか k 個を求める k-Nearest Neighbor Query が多く用いられている。しかし、このような類似検索の方法ではデータ空間が高次元になるとデータが疎になるため、R-Tree などの索引木では節同士の重なりが増すという意味で検索時のコストが増すという問題点があることが近年の研究で明らかになっている。[4]

本研究では、上記の問題点の解決のために近傍連鎖点列という概念を導入し、その差分更新の方法について述べる。近傍連鎖点列ではあらかじめ静的な索引付けが可能であることから高次元であることが検索時の直接のコスト増大につながらないため、その可用性を増すために静的索引の差分更新について述べていくことにする。

2 近傍連鎖点列

近傍連鎖点列 (Chained Neighborhood Points)* とは、多次元空間内に格納されている各点からの最近傍点を順次連結したものである。n 次元空間内の点の集合 S 内の任意の 2 点を o_i,o_j 間の距離を d_{ij} で表すとき、点 o_1 を起点とする近傍連鎖点列 (CNP_1) を以下のように定義する。

$$CNP_1 = \{o_1 \xrightarrow{d_{1,2}} o_2 \xrightarrow{d_{2,3}} \ldots \longrightarrow o_{d-1} \xrightarrow{d_{(d-1),d}} o_d\} \quad (1)$$

(但し、$o_j = \{o_i \in S | \forall p \in S : d(o_i, o_j) \leq d(o_i, p)\}$)

右向きの矢印 (→) は両端の 2 点の隣接関係を表し、左辺の点の最近傍点が右辺の点であることを表す。最近傍点の連結であることから、CNP の要素間の距離は単調減少するという性質を持つ。また、o_1 を起点オブジェクト、o_d を終点オブジェクトと呼ぶ。

2.1 CNP の索引方法

CNP の静的な索引方法には検索時索引と静的索引の 2 通りが存在する。検索時索引とは、名前の通り検索時に与えられた質問点からの最近傍点を順次連結して行くものである。検索時索引の際には、一度検索された点の CNP をデータベースに保存しておくキャッシュも利用できる。静的索引とは多次元空間内の全ての点に対し、あらかじめ CNP を生成しておく方法である。検索時には質問点からの k-Nearest Neighbor を求め、得られた点からの CNP を検索結果とする。

静的索引を生成できるという点は、CNP の大きな利点であり、高次元での検索コストの増大を回避する一つの解決策であると考える。本稿では以降、静的索引の利用価値を上げるためにその差分更新の方法について述べる。

3 CNP の差分更新

CNP の差分更新について述べる前に、静的な索引の格納方法について述べる。静的な索引の格納方法には、

(a) 検索時索引
　各点の最近傍点を求め、それを保持しておく。

(b) 静的索引
　起点オブジェクトから終点オブジェクトまでを保持しておく。

の 2 通りがある。以下ではそれぞれの静的索引について差分更新の方法を述べて行く。

3.1 最近傍点のみを保持する場合

まず (1) の最近傍点のみを保持する場合について述べる。図 1 の点 i が新しく挿入されるものとする。細い矢印は挿入前の CNP 太い矢印は挿入後に新しく加わった CNP である。図 1 では、挿入前は $a \to b \to c \to d$ と $e \to f \to g \to h$ の 2 つの CNP だったが、点 i の挿入後は $a \to b \to i \to c \to d$ と $e \to f \to i \to c \to d$、$g \to h$ の 3 つの CNP になっている。

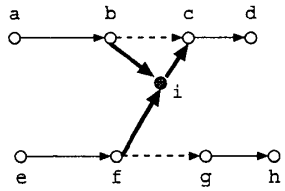

図 1: データの挿入

SATO Yasuhiro†, ITO Akinori†, TOYAMA Motomichi‡
†School of Science for Open and Environmental Systems, Faculty of Science and Technology, Keio University.
‡Department of Information and Computer Science, Faculty of Science and Technology, Keio University.

* 以下 CNP と略記する。

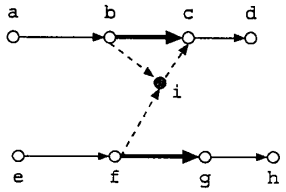

図 2: データの削除

挿入時の差分更新のアルゴリズムは、

ai1 データを挿入する。

ai2 挿入点の最近傍点を求め、格納する。

ai3 挿入点が最近傍点となる点を検索する。

ai4 step ai3 で得られた点を更新する。

である。step 3 で挿入点が最近傍となる点を検索する際に検索漏れをなくすために以下の定理を導入する。

定理 1 挿入点を I、点 I の k-近傍点のうちの 2 点を A, B とする。$\angle AIB$ を θ とすると、点 I が $\theta < \frac{2}{3}\pi$ となるような点 A,B で周り全てを囲まれるとき、点 A,B のような点の集合を S とすると、I が最近傍点となるような点は集合 S の要素中にしか存在しない。

定理 1 は 2 次元の場合であるが、より高次元の場合に関しても同様の定理が成り立つと考えられる。以下簡単のため 2 次元でのアルゴリズムについて述べる。定理 1 より∠AIB は $\theta < \frac{2}{3}$ であるから、挿入点 I を囲むには最低 4 つの三角形が必要となる。このことと定理 1 を利用して、挿入の step 3 を細かくすると、

ai3-1 k を 4 として、挿入点から k-近傍探索を行う。

ai3-2 step ai3-1 で得られた 4 点と挿入点で三角形をつくり、角度 θ をそれぞれ求める。

ai3-3 θ が全て $\frac{2}{3}\pi$ より小さければ探索を終了。$\frac{2}{3}\pi$ 以上の θ が存在する場合は、k を 1 増やして再度 k-近傍探索を行う。

ai3-4 θ が全て $\frac{2}{3}\pi$ より小さくなるまで、step ai3-2 と ai3-3 を繰り返す。

となる。step ai3-1 から ai3-4 を行うことで、新しい点に影響される点の更新漏れを防ぐことができる。同様にデータを削除する場合の差分更新は、

ad1 データを削除する。

ad2 削除した点が最近傍点であった点を索引から検索する。

ad3 step ad2 で得られた点から最近傍探索を行い、索引を更新する。

となる。図 2 で点 i が削除されるとすると、点 i が最近傍点だった b, f の最近傍点をそれぞれ c, g に変更し、$a \to b \to c \to d$ と $e \to f \to g \to h$ の 2 つの CNP のみになる。

3.2 起点から終点オブジェクトまでを保持する場合

CNP の起点オブジェクトから終点オブジェクトまでを格納する場合の差分更新方法も、基本的には最近傍点のみを格納する場合と変わらない。挿入する場合の差分更新方法は、

bi1 データを挿入する。

bi2 先の step ai3-1 から ai3-4 を実行し、挿入点が最近傍点となる点を求める。

bi3 step bi2 で得られた点を含む CNP を索引から検索する。

bi4 step bi3 で得られた CNP を更新する。

となる。最近傍点のみを格納する場合に比べて、対象となる CNP そのものを更新するため、更新の際の最近傍探索の回数が多く更新のコストは大きくなる。しかし検索時には索引への問合せが一度で済むため検索のコストは小さい。

削除の場合の差分更新も基本的には最近傍点のみを格納する場合と変わらない。

bd1 データを削除する。

bd2 step ai3-1 から ai3-4 を実行し、削除した点が最近傍点である点を求める。

bd3 step bd2 で得られた点を含む CNP を索引から検索する。

bd4 step bd3 で得られた CNP を更新する。

step bd4 で更新する際、削除されたことにより終点オブジェクトとなった点から更新する。終点オブジェクトとなった点以外は最近傍点が変わることはないので更新する必要がない。図 2 では、点 b, f は点 i を最近傍点としていた点であるだけでなく、$a \to b$ と $e \to f$ の終点オブジェクトであるために点 b, f から先の CNP を更新する必要がある。

4 まとめと課題

多次元データの検索手法として近傍連鎖点列の概念を採用し、その静的な索引の差分更新の方法について述べた。静的な索引を用いることで高次元での検索コストの増大の問題を回避することができる。今後の課題としては、今回述べなかった k-近傍点を順次連結する CNP 拡張形の場合の差分更新について検討して行きたいと考えている。

参考文献

[1] 田中 覚, 遠山 元道: 多次元空間における近傍連鎖点列を用いた類似検索システム, Database Engineering Workshop 2001, 4B-3.

[2] 佐藤 康裕, 田中 覚, 遠山 元道: 近傍連鎖点列における静的索引, Database Workshop 2001, 5X-3.

[3] 佐藤 康裕, 伊藤 晶規, 田中 覚, 遠山 元道: 近傍連鎖点列の概念に基づく画像検索システムの実装と評価, Database Engineering Workshop 2002, 1B-5.

[4] Kevin S. Beyer,Jonathan Goldstein,Raghu Ramakrishnan,Uri Shaft: When Is "Nearest Neighbor" Meaningful?, ICDT 1999, pp217-235.