移動オブジェクトを管理する時空間データ構造

正員 出木原 裕順 (広島国際大学)、非会員 中村 泰明 (広島市立大学)

An Efficient Spatial-Temporal Data Structure for Moving Objects

Hiroyuki Dekihara, Member (Hiroshima International University)

Yasuaki Nakamura, Non-Member (Hiroshima City University)

A spatial-temporal data structure, called the PMD-tree (Persistent Multi-Dimensional tree), has been proposed for managing the live intervals and locations of spatial objects. In the paper, novel concepts of time space bounding box (TSBB) and motion list are introduced to the PMD-tree to manage moving spatial objects efficiently. A TSBB is an extended bounding box for a moving object that covers the trajectory of the object. As an object moves, a TSBB corresponding to the object is enlarged to enclose the trajectory of the object. A TSBB is divided, when it becomes greater than a limit. An object and corresponding TSBBs are managed by a doubly connected linked list, called motion list. TSBBs are also managed by the PMD-tree. Introducing the concept of TSBB and motion list to the PMD-tree, moving objects can be efficiently managed and be quickly found for spatial-temporal queries. By the series of simulation tests, the storage requirements and search performances are evaluated for several types of moving objects. As a result, the proposed method is superior to the conventional methods.

キーワード・時空間データ、時空間データ構造、移動データ

Keywords : spatial-temporal data, spatial-temporal data structure, moving object

1. はじめに

モニタリングシステムや緊急車両などの指揮システムなどでは、静止したデータはもちろん、移動するデータも効率的に管理できるデータ構造が要求される。また、コンピュータ・グラフィックスやバーチャル・リアリティなどの分野でも、動画やアニメーションなどの膨大なデータを管理するために、移動するデータの効率的な管理が可能なデータ構造が要求される。移動するデータとは、例えば、時刻 t_1 の x - y 座標から時刻 t_2 の x - y 座標へ座標が変わる様に、時間経過に伴い座標位置が変化するデータ（以下、移動オブジェクトと呼ぶ）のことであり、空間情報（ x - y 座標）+時間情報で表現される時空間データの一種と言える。従って、移動オブジェクトを効率的に管理するためには、時空間データ構造を用いる必要がある。

近年、多くの時空間データ構造が開発されている[1-5]。特に、データ空間を再帰的に分割し、分割された領域を各葉に対応させて管理する木構造を応用したデータ構造が数多く提案されている[2-5]。著者らも、1997年に、時空間データを効率的に管理するため、時間経過に伴う木構造の変化の差分を保持する手法を用いて、多次元完全平衡木である空間データ構造：MD 木[6]の拡張を行い、時空間データ構造：

Persistent MD 木 (PMD 木) [7,8]を開発した。PMD 木は、各時刻に対するデータ構造における管理構造の時間方向の差分のみを保持して木構造の圧縮を行っている。一方、MPEG などでも、時間方向の画像の差分に基づいてデータの圧縮を行っている。

しかし、それらのデータ構造[2-8]で移動オブジェクトを管理しようとした場合、移動前後のオブジェクトのデータはそれぞれ別のデータとして管理され、新しいデータの追加による点情報によって、オブジェクトの移動の軌跡を近似するため、オブジェクトの移動に比例してデータ数が増加してしまい、オブジェクトの管理に多大なコストを要するか、もしくは、不可能であるものがほとんどであった。

本研究では、Time Space Bounding Box (TSBB) とモーションリストという2つの新しい手法を考え、PMD 木に適用して移動オブジェクトの管理が可能となるよう拡張を行う。TSBB は、Bounding Box (BB) と呼ばれる、各オブジェクトデータが保持しているオブジェクト自身を包含する $x \cdot y$ 軸に平行な辺を持つ長方形（データの BB）、およびノード・葉が管理している全データを包含する $x \cdot y$ 軸に平行な長方形（ノード・葉の BB）を、移動データが描く軌跡も包含するように拡張したものである。

ここで、オブジェクトの移動に伴い TSBB が拡大してい

った場合、無制限に大きくなりすぎてしまうと、データ構造のパフォーマンスが低下してしまう。例えば、あるオブジェクトが移動した結果、このオブジェクトの TSBB の大きさが全データ空間と同じ大きさになってしまった場合、このオブジェクトは全データ空間上に存在しているものとみなされる。従って、このオブジェクトを管理しているノードと葉の TSBB も全データ空間と同様の大きさになり、これらの部分木とオブジェクトが格納されている葉、および、移動したオブジェクトは、データ検索時において、常に検索対象として検出されてしまう。なぜならば、全データ空間と同等の大きさになった各 TSBB 内に、Query として与えられる検索範囲が常に包含されるため、それらの TSBB 内に求めるオブジェクトが存在する可能性があると判断され、葉のデータを 1 つずつ調べる必要が生じてしまうからである。このように、TSBB が無制限に拡大すると、データ空間を木構造によってデータ空間を分割した利点がなくなってしまい、パフォーマンスが低下してしまう。

この TSBB の拡大におけるパフォーマンスの低下を防ぐために、ある制限を与え、この制限を越える場合には、TSBB の分割を行い、TSBB の拡大問題を解決する。制限には、①移動オブジェクトの移動距離が基準値以上になる、②TSBB の一辺の長さが基準値以上になる、③TSBB が現在属している葉の管理領域の境界線を越える、の 3 つが挙げられる。これらの制限で分割された各 TSBB をトレースと呼ばれるデータ構造で管理する。トレースでは、分割された区間に対するデータの移動情報などの管理を行う。それらのトレースを双方向リストで結んだリストをモーションリストと呼び、データの全移動情報の管理を行う。

この 2 つの手法を用いて、PMD 木の拡張を行い、移動データの管理が可能な新しい時空間データ構造を提案する。また、実際にシステムを実装してシミュレーション実験を行い、従来法との比較評価を行う。従来法で最も多く採用されている、1 つの木構造ですべての時刻のデータを管理する手法により拡張した MD 木と TSBB とモーションリストを使用しない PMD 木を従来法として取り上げ比較実験を行い、その有用性を示す。なお、それぞれのデータ構造において、移動データは、データの挿入による近似法を採用する。また、TSBB の分割における各制限に対する性能評価を行い、TSBB の分割条件について考察する。

2. 時空間データ構造 –PMD 木–

PMD 木は、多次元完全平衡木である空間データ構造 MD 木[6]に、時間経過による木構造の変化を保持する手法を用いて拡張した時空間データ構造である。

〈2・1〉 MD 木による空間データの管理 木構造を用いて空間を管理する場合、一般的に、空間を再帰的に分割し、分割された領域を各葉に対応させて管理する手法を用いたデータ構造が、その有用性により、数多くのシステムで採用されている。以下、MD 木を例に挙げて空間データ構造の説明を行う (図 1 参照)。MD 木の内部ノードは、そのノード以

下の子 (葉・ノード) が持つすべての管理領域を包含する x y 軸に平行な最小の長方形を管理領域として持っている。図中の木構造内の灰色部分が管理領域である。なお、データは各葉に格納されている。

また、図 1 中で、破線で描かれた長方形を Bounding Box (BB) と呼び、各ノード・葉が管理している全データを包含する x y 軸に平行な最小の長方形で表す。BB は、検索操作の高速化に寄与している。例えば、図 1 の Query Search Area (検索範囲) の範囲をデータ検索する場合、検索対象となる葉は、葉の管理領域と検索範囲が重複している L1、L2、L4 の 3 つがまず上げられる。ここで、更に葉の BB を調べることで、検索対象となる葉の選別が可能である。葉の BB は葉で管理されている全データを包含する長方形である。従って、その長方形内には必ずデータが存在している。逆に、その長方形外にはデータは必ず存在しない。よって、図 1 より、検索範囲と、L1、L2、L4 の BB との重複を調べると、L4 の BB は検索範囲と重複していないため、L4 を検索対象から除くことができ、L1 と L2 のみ検索すればよい。このように、BB はデータ検索時に疎な検索を行うことにより、検索操作の高速化を行っている。また、空間的に広がりを持つ図形データを管理する場合には、データを包含する最小の長方形を図形データの BB として定義し、検索時に疎な検索を行ってデータを検出する。例えば、丸型の図形データを管理する場合、検索範囲と丸型データの曲線との重なりを個別に判定してデータを検索するよりも、データの BB を調べて疎な検索を行い、候補の選出を行ってから詳細な判別を行う方が実用的であり、かつ、効率的である。これらの機能は、拡張された PMD 木も保持している。

図 1 MD木による空間データ管理

Fig.1 An example of MD-tree

〈2・2〉 PMD 木の基本構造と構築の流れ PMD 木は、MD 木を拡張した図 2 (a) のような根付き木構造であり、空間を再帰的に分割し、データの時間毎の処理 (追加・削除) に伴うデータ構造の変化の差分を保持することにより、現在の状態だけでなく過去の状態へのアクセスも可能な時空間データ構造に拡張されている。葉とノードには現在有効であるデータを保持しているもの (現存葉・現存ノード) と過去に有

効であったデータを保持しているもの(保存葉・保存ノード)の2つがある。

図2 (a) の白塗りの部分が現在有効であるデータを管理している部分で、黒塗りの部分が過去のデータの状態を管理している部分である。ノードにはそのノードが作成された時刻 (Ts) と保存ノードになった時刻 (Te) を存続期間 [Ts, Te) として保持している。また、データも同様に、追加された時刻 (Ts) と削除されるまでの時刻 (Te) をデータの有効時間 [Ts, Te) として持っている。各データは葉に格納される。また、内部ノードは2~4個の子ノードを持つことができ、木の終端ノードは2~3個の葉を持つことができる。各ノードは過去の状態へのポインタ (旧ノードへのポインタ) を持つ。

(b) 分割されたデータ空間の管理

図2 PMD木の構造と分割空間の管理

Fig2. Examples of PMD-tree and management of data-space

図2 (b) により、データ空間の分割管理を説明する。分割が発生する以前はノード A1・A2 で全データ空間を管理している。データがノード A1 の管理領域に投入されると、やがて葉の容量が一杯になる。その状態から更に A1 にデータが投入されると分割が発生する。A1 の管理領域が分割されると、A1 の領域を、新しく作成したノード B1・B2 の2つのノードで管理し、前の状態 (A1) は過去の状態として保存しておく。ノード A1 で管理していた領域をノード B1・B2 の2つのノードで管理することから、この分割をノードの分割と呼ぶ。分割後は、データ空間は A2・B1・B2 の3つのノードで管理される。なお、ノードの分割については、文献[7,8]に詳しい。

図3に PMD 木の構築の流れを示す。PMD 木に最初のデータが投入されると、根 R0 とデータを格納する葉が1つ作られる。さらにデータが追加されると、葉の分割が発生していき、やがて葉が3つとなり、R0 が満杯の状態になる (図3 (a) 左図参照)。なお、葉の分割に関しても文献[7,8]に詳しい。その状態で新しいデータが追加されると、ノードの分割が発生し、新しい根 R1 とノード A1・A2 が作成される (図3 (a) 右図参照)。A1・A2 は、それぞれ旧ノードへのポインタを保持している。

図3 PMD木の構築の流れ

Fig.3 Growing processes of PMD-tree

ここで、A1 の管理領域にデータが追加されていくと、葉の分割が発生し、葉の数が3つになり、やがてノードの分割が発生し、新たにノード B1・B2 が作成される。B1・B2 は R1 によって管理される。この時、PMD 木の構造は、図2 (a) の様な木構造になる。

図2 (a) のデータ構造に対して、ノード A2 にデータが追加され、ノードの分割が発生した状態が図3 (b) である。ノードの分割により新しいノード C1・C2 が作成されるが R1 はこれ以上子ノードを持つことができないので、新しい根 R2 と内部ノード I1・I2 が作成され、これらによってデータ空間が新しく管理される。PMD 木の根は根管理テーブルによって存続時間とともに管理される (図3 (b) 参照)。

一般に、MD 木や PMD 木などの木構造では検索時間は木の深さに比例する。PMD 木はすべての時刻の状態を保持しているにもかかわらず、各時刻に対する木構造の変化の差分を管理する手法により、PMD 木の各時刻に対する部分木の深さは、各時刻に有効であった (削除されていない) データのみに依存する。しかし、全ての時刻のデータを単一のデータ構造で管理する従来法では、木の深さは全ての時刻の総データ数に依存してしまう。従って、全データを単一のデータ

構造で管理する従来法よりも、木構造の差分を管理する PMD 木は木の深さが浅くなり、各時刻のデータへの効率的なアクセスが可能である。これら PMD 木の特徴については文献[7,8]に詳しい。

〈2・3〉 過去のデータへのアクセス例 図 3 (b) より、時刻 $T1 \sim T3$ までにデータ構造内に存在していたデータに対してアクセスする場合、まず、根管理テーブルより時刻 $T1 \sim T3$ に対応する根 $R1$ を選出する。次に、 $R1$ のポインタをたどっていき、 $R1$ に対する部分木のノードの存続期間と管理領域を参照しながら時空間探索を行っていく。図 3 (b) の点線で囲まれた部分木のみが検索対象となる。この時、必要があれば、旧ノードへのポインタをたどって過去のノードを探索する。

3. 移動オブジェクトの管理

移動オブジェクトを管理するために、Bounding box (BB) を拡張した Time Space Bounding Box (TSBB) と各 TSBB を管理するトレースと呼ばれるデータ構造を双方向で連結したモーションリストの2つの新しい手法を用いて PMD 木の拡張を行う。

〈3・1〉 Time Space Bounding Box (TSBB) BB は、各辺が軸に平行で、かつ、管理している全データを包含する最小の長方形のことであり、空間検索の高速化に寄与する (2・1 節参照)。従来の PMD 木において、BB は、ノード・葉がそれぞれ保持している。ただし、空間的な広がりを持ったオブジェクトを管理する場合には、オブジェクト自身を包含する BB をデータの BB として管理していた。移動オブジェクトを管理するにあたって、オブジェクトが移動したときの軌跡も包含する様に BB を拡張したものが TSBB である。TSBB は、ノード・葉・オブジェクトデータのそれぞれに保持させる。図 4 より、移動オブジェクトが移動した場合、移動したオブジェクトの軌跡も含むように、そのオブジェクトの TSBB が拡大される (図 4 (a) 参照)。

図 4 移動に伴うノード・葉・データの TSBB の拡張
Fig.4 Extended TSBB of nodes, leaves and moving data

その時、そのオブジェクトが属している葉 $L4$ の TSBB の外に移動オブジェクトが移動するならば、 $L4$ の TSBB も拡大させる (図 4 (b) 参照)。葉の TSBB が変化した場合、それに合わせてその葉を管理している内部ノードや根の TSBB を変化させる必要がある。従って、 $L4$ の TSBB が変化したため、内部ノード $A2$ と根 R の TSBB を変化させる。変化後の木構造を図 4 (c) に示す。

〈3・2〉 モーションリスト オブジェクトが時間経過に伴って移動し続けた場合、TSBB が限りなく大きくなってしまうと、データ構造の性能が著しく低下してしまう。仮に、あるオブジェクトの TSBB が全データ領域と同面積になってしまった場合、そのオブジェクトを管理しているすべてのノード・葉の TSBB が全データ領域と同じ大きさになってしまう。従って、データアクセス時にそれらは常に検索対象として選出されてしまい、データを絞り込んでいくことが困難になる。これでは空間を再帰的に分割した利点が無くなってしまう。そこで、TSBB を限りなく拡大するのではなく、制限を設定し、その条件を満たす場合には、TSBB の分割を行い、この TSBB の拡大問題を解決する。分割された TSBB の空間情報と分割時刻などの時間情報は、トレースと呼ばれる、分割区間の時空間情報を管理するデータ構造によって管理される。それらのトレースを双方向で連結したリストでオブジェクトの移動を管理する。このリストをモーションリストと呼ぶ (図 5 参照)

図 5 モーション・リストの例
Fig.5 A Example of Motion List

図 5 より、オブジェクトが時刻 $T1$ から $T4$ まで移動したとき、本来ならば図 5 左図の様に、TSBB が拡大される (なお、 $(x(Ti), y(Ti), Ti)$ は時刻 Ti におけるデータの座標 $(x(Ti), y(Ti))$ を表す)。しかし、TSBB の分割条件により、時刻 $T2$ と $T3$ で分割が発生した場合、図 5 右図の様に TSBB が分割され、分割された各 TSBB をトレースで管理し、それらを双方向リストで連結する。これがモーションリストである。トレース $T1$ の管理している時間間隔は $T1 \leq t < T2$ であり、トレース $T2$ は $T2 \leq t < T3$ 、トレース $T3$ は $T3 \leq t < T4$ である。従って、モーションリストは、 $T1$ から $T4$ までのデータの移動を管理することになる。TSBB の分割の条件としては、以下の 3 つの方式を設定し、シミュレーション実験によって比較評価を行う。

- ① 移動オブジェクトの移動距離が基準値以上になる。
- ② TSBB の一辺の長さが基準値以上になる。

- ③ TSBB が現在属している葉の管理領域を越えて拡大し、管理領域の境界線を越えた TSBB の範囲が基準値以上になる。

トレースを用いたデータ管理の流れを図 6 に示す。時刻 T_i に移動オブジェクトが座標 $(x(T_i), y(T_i))$ に投入された場合、オブジェクトを包含する TSBB とそのオブジェクトを管理するための基準点 $(x(T_i), y(T_i))$ を $\text{Trace}(T_i, -)$ で管理する。この時、データの軌跡情報 (locus) は Empty である。なお、時刻 T_i から T_j までを管理しているトレースを $\text{Trace}(T_i, T_j)$ と記述し、同様に、時刻 T_i から T_j までを管理している TSBB を $\text{TSBB}(T_i, T_j)$ と記述する。オブジェクトが移動すると、TSBB がオブジェクト自身とオブジェクトの軌跡を包含するよう拡大されていく。分割条件により、時刻 T_{i+1} において TSBB が分割される場合、その時の時刻 T_{i+1} を終了時刻として、前のトレースは保存状態となり、新しいトレース $(T_{i+1}, -)$ で、新しい基準点 $(x(T_{i+1}), y(T_{i+1}))$ とオブジェクトを包含する TSBB を共に管理する。更にデータが移動していった場合、TSBB と locus をそれに合わせて調整し管理していく。また、図 6 より、新しい Trace が作成された際、locus が empty 状態であるのは、そのデータに軌跡情報が存在しない状態を表す。なお、Trace の構造内では locus へのポインターが NULL になっており、余分なデータ量が増えることはない。

図 6 トレースによるデータ管理の流れ

Fig. 6 Management moving object by traces

また、1 つのトレースは、分割されない限りデータの移動軌跡を蓄えて管理している。例えば、あるデータ A が、時刻 T_1 より移動した場合に、TSBB の分割が生じたのが時刻 T_{10} であったと仮定すると、時刻 T_1 から時刻 T_{10} までの間の軌跡情報は、1 つのトレース (T_1, T_{10}) で管理される。もし、TSBB の分割に基づくトレースの管理がなく、時刻 T_1 から時刻 T_{10} の間を従来法のように、各時刻に新しい

データを挿入して、それらのデータ群で近似して管理した場合、新しいデータが 9 個挿入され、10 個の点で軌跡情報が近似される。言い換えると、トレース (T_1, T_{10}) はこの 10 個の点情報で近似されたものと、同等の軌跡情報を保持している。

〈3・3〉 システムの概要とデータ検索 システムの概要は図 7 の様になり、PMD 木とオブジェクト DB とトレース DB の 3 つから構成される。オブジェクトデータの属性 (色や材質や作成時刻などの時間情報といったオブジェクト固有の情報)、形状 (オブジェクトの空間的な情報)、機能 (移動ベクトルの方向や速さなどの動作情報) などの詳細な情報はオブジェクト自身がパラメータや数式などの形式により保持し、それらはオブジェクト DB に蓄えられる。また、オブジェクトの移動した情報は、オブジェクトとオブジェクトの軌跡を包含する TSBB を分割してトレースで管理を行い、それらを双方向リストで連結したモーションリストによってデータの移動情報の管理が行われ、モーションリストは、トレース DB で蓄えられる。PMD 木では、データ空間の再帰的な分割による空間情報のカテゴリー化とデータ構造の差分を用いた時間情報のカテゴリー化による、時空間データの Kategorization を行い、時間情報と空間情報による効率的なアクセスを可能にしている。

あるオブジェクトデータを検索する場合、まず、PMD 木を用いて時間情報と空間情報から検索対象と考えられるデータを検出する。これらのデータは、TSBB での検索によって選ばれた、言い換えると、疎な検索によって選ばれたデータであり、まだ完全に検索対象とは言えない。従って、選出されたデータを詳細に調査するため、トレース DB 内にある各データのモーションリストを探索し、TSBB 内のオブジェクトの軌跡を調べていく。データの形状など、個別の情報によって検索対象を確定する場合には、更にオブジェクト DB を調べることにより検索結果を導出する。

図 7 システムの概要

Fig. 7 Design of systems

4. シミュレーション実験

提案するデータ構造 (以下、Extended PMD 木と呼ぶ) と、従来法との比較評価実験を行う。実験では、地図情報や設備情報などの静的なデータの中を、移動オブジェクトが移動する場合を想定する。また、従来法に最も採用されている、

1 つのデータ構造ですべての時刻のデータを管理する手法、および、データの移動に伴ってデータを挿入し、それらのデータ群により移動データの軌跡情報を近似して管理する手法を用いて改良した MD 木と、改良した MD 木と同様に、データの移動情報はデータ追加による近似で移動オブジェクトを管理する PMD 木を取り上げ、提案するデータ構造との比較評価を行う。なお、データ間の詳細な軌跡情報は、パラメータや数式などの形式で各データ自身が保持するものとする。

提案するデータ構造の TSBB の分割条件としては、①移動オブジェクトの移動距離が基準値 L 以上になる、②TSBB の一辺の長さが基準値 H 以上になる、③TSBB が現在属している葉の管理領域を越えて拡大し、管理領域の境界線を越えた TSBB の範囲が基準値 C 以上になる、の 3 つとし、それぞれを比較して性能評価を行う。また、葉の管理領域を越える TSBB の許される範囲を TSBB の許容範囲 (T) とし、葉の数を L とすると、 $T = k \cdot 1/\sqrt{L}$ ($0.1 \leq k \leq 1$) という式により許容範囲 (T) を定義する。本実験では、 L H は、(全データ空間の短辺の長さ) $\times 0.01$ とし、 C は、(葉の管理領域) $\times 1.2$ とする。

〈4・1〉 シミュレーション条件 時間情報をデータ属性として保持する 2 次元点データを移動オブジェクトデータとする。オブジェクトの位置 (X, Y) は一様乱数により発生させる。ただし、全データ空間は正方形であり、1 辺の長さは 10,000 以下とし、データの追加範囲は $0 \leq X \leq 4,096$ 、 $0 \leq Y \leq 4,096$ とする。従って、TSBB の分割条件における L と H は共に 100 になる。時刻 0 に {5,000・10,000・50,000・100,000} 個の初期データをデータ構造に投入し、その後、1 単位時間が経過するごとに 10% の移動オブジェクトがランダムで移動するものとし、100 単位時間経過 (終了時刻 100) させる。オブジェクトの移動速度 V は、(V_x , V_y) のパラメータで定義し、 $|V_x| < 100$ 、 $|V_y| < 100$ とする。100 単位時間経過後、時刻と時間間隔を指定した時空間検索を行う。なお、それぞれのデータ構造において葉に格納できるデータの数は 20 とする。また、検索領域は、全データ空間を 1 としたときの面積の割合 (%) で表し、計算機は、DEC ALPHA STATION 500/500 を使用した。

〈4・2〉 実験結果 表 1 と表 2 に、Extended PMD 木と MD 木の実験後のデータ構造を示す。なお、表 1 における Extended PMD 木の TSBB の分割条件には、分割条件①(データの移動距離 $\geq L$) を採用している。また、従来法であるデータの挿入による軌跡近似の手法を用いた PMD 木の結果を表 3 に示す。CPU 時間は、移動オブジェクトを 100 単位時間移動させたとき、データ構造がそれに対応して構築されるのに費やした時間を表す。

表 1 と表 2 より、オブジェクトの移動によるデータ構造の構築時間は Extended PMD 木の方が MD 木より、6~27%程度速くなっている。また、データ構造に必要なノードと葉のメモリ数 (No. of nodes, No. of leaves)、および、葉で管理されるデータ数 (No. of total data) においても、

Extended PMD 木の方が、41~57%・30~45%・76~79%程度減少している。なお、葉で管理されるデータ数では、トレース情報も含まれたデータ数をカウントしている。

また、表 1 と表 3 より、提案する Extended PMD 木と、TSBB とモーションリストを使用せず、データの移動に伴って新しいデータを挿入し、それらのデータ群によって軌跡情報を近似して管理する従来法を用いた PMD 木の実験結果を示す。TSBB とモーションリストを使用しない従来法による PMD 木は、データ構造に必要なノードと葉のメモリ数、および、葉で管理されるデータ総数において、Extended PMD 木の 4.32~6.25 倍・4.48~6.31 倍・4.10~4.61 倍程度ものメモリ数が必要となる。また、データ構造の構築時間においても、2.13~4.34 倍程度必要になっている。

表 2 と表 3 より、従来方による PMD 木と MD 木を比較しても、データ構造に必要なノードと葉のメモリ数では、MD 木の 2.26~3.65 倍・3.04~4.69 倍程度のメモリ数がデータ構造に必要であり、データ構造の構築時間においても 1.97~3.23 倍程度必要になる。よって、TSBB とモーションリストを使用しない PMD 木は、データ構造に必要なコストが高く実用的ではなく、従来法よりも TSBB とモーションリストを用いて移動するデータを PMD 木で管理する方が効率的であることが確認できる。

表 1 分割条件① (データの移動距離 $\geq L$) に対する

Extended PMD 木のデータ構造

Table 1. Extended PMD-tree of construction in the once rule of splitting TSBB(distance of data $\geq L$)

No. of initial data	No. of nodes	No. of leaves	No. of total data	CPU time (μ sec)
5,000	1,179	2,070	11,939	0.53
10,000	2,815	4,588	26,850	1.28
50,000	10,341	18,085	116,731	9.53
100,000	20,914	37,374	238,154	22.75

表 2 MD 木のデータ構造

Table 2. MD-tree of construction

No. of initial data	No. of nodes	No. of leaves	No. of total data	CPU time (μ sec)
5,000	2,383	3,268	55,000	0.72
10,000	4,778	6,581	110,000	1.75
50,000	24,022	32,759	550,000	11.33
100,000	47,886	65,458	1,100,000	24.17

表 3 PMD 木のデータ構造

Table 3. PMD-tree of construction

No. of initial data	No. of nodes	No. of leaves	No. of total data	CPU time (μ sec)
5,000	8,688	15,317	55,000	2.32
10,000	14,199	26,100	110,000	4.55
50,000	54,327	99,599	550,000	22.35
100,000	109,136	199,383	1,100,000	48.43

初期データ 100,000 に対して、検索時刻を指定した時空間検索の結果を図 8 に示す。なお、検索範囲は、全データ空間に対する検索範囲の面積の割合 {1%・5%・9%} につ

図8 Extended PMD 木と MD 木の時間指定検索結果
Fig.8 Search performances of Extended PMD-tree and MD-tree (set a time)

図9 Extended PMD 木と MD 木の時間間隔指定検索結果
Fig.9 Search performances of Extended PMD-tree and MD-tree (set interval)

いて実験を行った。図8より、Extended PMD 木の検索時刻を指定した時空間検索時に対象となるノード数は、MD 木に比べ、67~85%程度減少しており、検索時間において、3.35~9.65 倍程度高速化されている。

また、初期データ 100,000 に対して、時間間隔を指定した時空間検索の結果を図9に示す。なお、検索範囲は1%に固定し、現在時刻(100)から過去に向けて時間間隔を広げていき、時間間隔 {90~100・80~100・60~100・40~100・20~100・0~100} を指定して時空間検索を行った。図9より、時間間隔を指定した時空間検索において、Extended PMD 木は、検索時間において、MD 木の 1.03~2.67 倍程度高速化されている。また、検索時に対象となるノード数においても、時間間隔が全体の約 80%に達するまで、最悪でも MD 木と同等、もしくは最善で 59%程度減少している。検索範囲が全体に及んだとしても Extended PMD 木は MD 木と比べて 10%程度しか探索対象ノード数が増加しておらず、検索性能が低下していない。以上より、提案する Extended PMD 木は、効率的に移動オブジェクトを管理するデータ構造であると言える。

また、表4と表5にTSBBの分割条件②(TSBBの一边の長さ $\geq H$)と③(葉の管理領域を越えたTSBBの範囲 $\geq C$)におけるExtended PMD 木の実験後のデータ構造を示し、表1(分割条件①)と表4(分割条件②)、および、表5(分割条件③)より、3つの分割条件によるExtended PMD 木の比較評価を行う。表1・表4より、分割条件①よりも分割条件②のデータ構造の方が必要なメモリ数が少なくて済む。なぜならば、分割条件①では、データが一定距離移動すれば必ず条件に当てはまり、TSBBの分割が発生する。しかし、分割条件②では、現在の大きさのTSBB外にデータが移動し、かつ、一边の長さが基準値H以上にならなければ分割が発生しないため、分割条件①より分割条件②の方がデータ数の増加が抑えられて効率的になる。また、表5より、分割条件③においては、データ構造で管理するデータ数が少ない場合、言い換えると、葉の管理領域の面積が大きい場合は、データが葉の管理領域内を移動する可能性が高いため、TSBBの分割が発生しにくく、データ構造内で管理されるデータ数は少なく、システムに必要なメモリ数も小さくなる。よって、TSBBの分割条件については、管理する移動オブジェクトの性質・環境に合わせて使用することにより、より効率的な移動オブジェクトの管理が可能である。

表4 分割条件②(TSBBの一边の長さ $\geq H$)に対する
Extended PMD 木のデータ構造

Table 4. Extended PMD-tree of construction in the second rule of splitting TSBB(TSBB a length $\geq H$)

No. of initial data	No. of nodes	No. of leaves	No. of total data	CPU time (μ sec)
5,000	930	1,597	9,201	0.52
10,000	2,030	3,442	19,984	1.15
50,000	6,783	11,963	82,380	9.02
100,000	14,901	26,264	171,655	19.68

表 5 分割条件③ (葉の管理領域を越えた TSBB の範囲 $\geq C$) に対する Extended PMD 木のデータ構造

Table 5. Extended PMD-tree of construction in the third rule of splitting TSBB(TSBB range over the managing area of leaves $\geq C$)

No. of initial data	No. of nodes	No. of leaves	No. of total data	CPU time (μ sec)
5,000	601	1,005	7,593	0.55
10,000	1,977	3,343	20,970	1.28
50,000	10,217	17,793	119,044	12.50
100,000	27,422	47,848	278,589	23.78

5. おわりに

PMD 木を拡張し、移動するオブジェクトの管理も可能な時空間データ構造を新しく開発した。また、シミュレーション実験を行い、従来法で拡張した MD 木との比較評価によってその有用性を示した。開発したデータ構造は、従来法に比べて、メモリ容量が 30~79%程度減少し、検索速度は 3.35~9.65 倍程度高速化されており、効率的なデータ構造であると言える。また、TSBB の分割条件における性能評価により、分割条件② (TSBB の一辺の長さ $\geq$ 基準値) の TSBB の分割条件が最もコンスタントで効率が良いと言える。しかし、データ数が少ない場合、すなわち、葉の管理領域の面積が大きい場合には、葉の管理領域外にデータが移動した場合に TSBB を分割する方が、すなわち、分割条件③ (葉の管理領域を越えた TSBB の範囲 $\geq$ 基準値) の方が効率的になる。従って、管理する移動オブジェクトの性質・環境などに合わせて、分割条件を適用すれば、より効率的な管理が可能である。

(平成 13 年 7 月 25 日受付, 平成 14 年 1 月 16 日再受付)

文 献

[1] H.Samet: Applications of Spatial Data Structures, Addison-Wesley (1989)

[2] J. R. Driscoll, N. Sarnak, D. D. Sleator, and R. E. Tarjan, "Making Data Structure Persistent," J. Comp. & Syst. Sci., Vol. 38, pp.86-124(1989).

[3] 仲、才脇、辻本、西田:「高速検索のための時空間データ管理方式—状態が複数の場合の検討—」信学技報 DE96-86, pp.73-78 (1997)

[4] E, Bertino, et. al. "Indexing Techniques For Advanced Database System", KAP, pp114-149(1997).

[5] V. J. Tsotras and B. Gopinath: Efficient Algorithms for Managing the History of Evolving Databases, Proc. Int'l Conf. Database Theory, 1990.

[6] 中村、阿部、大沢、坂内:「多次元データの平衡木による管理—MD木の提案—」, 電子情報通信学会論文誌 D, vol. J71-D, No.9, pp.1745-1752(1988).

[7] H. Dekihara and Y. Nakamura, "An Efficient Balanced Hierarchical Data Structure For Multiversion Accesses To Spatio-Temporal Data," Proc. EUSIPCO 98, pp.2205-2208(1998).

[8] 出木原、中村:「時間属性を持った空間データの管理構造 PMD 木」, 情処論, vol.40, No.SIG5(TOD2), pp.54-68(1999).

出木原 裕順

(正員) 1998 年広島市立大学情報数理学科卒業。2000 年同大学院修士課程修了 同年同大学院博士後期課程入学。2001 年広島国際大学助手, 現在に至る。画像処理, 時空間データ構造, 地理情報システムなどの研究に従事。電気学会, 電子情報通信学会各会員。

中村 泰明

(非会員) 1977 年大阪大学制御工学科卒業, 1979 年同博士課程前期修了 同年三菱電機株式会社入社。以来, 1994 年まで同中央研究所にて, 画像処理, CAD, パーチャルリアリティ, 空間データ構造, 地理情報システムなどの研究に従事。1994 年広島市立大学情報科学部情報数理学科教授。現在に至る。工学博士 (東京大学)。IEEE, 電子情報通信学会, 情報処理学会, 計測自動制御学会各会員。