移動軌跡ストリームからの移動統計量推定のための動的ヒストグラム構築手法について

塚本　祐一† 　石川　佳治†† 　北川　博之††

† 筑波大学システム情報工学研究科　〒305–8573 筑波大学つくば市天王台 1–1–1
†† 筑波大学電子・情報工学系　〒305–8573 茨城県つくば市天王台 1–1–1
E-mail: †yuichi@kde.is.tsukuba.ac.jp, ††{ishikawa,kitagawa}@is.tsukuba.ac.jp

あらまし　GPS や通信技術の発展に伴い，移動する多数の移動オブジェクトの移動状況の追跡が容易になりつつある．大量の移動オブジェクトの移動状況をリアルタイムに追跡し，分析・予測に役立てるには，ストリーム的に配信されてくる移動状況データを効率よく要約することが求められる．そこで本稿では，移動オブジェクトの移動軌跡データを効率よく集計する動的ヒストグラム構築手法を提案する．このヒストグラムはマルコフ連鎖モデルに基づく移動統計量に基づいており，OLAP 的な分析を支援する．

キーワード　移動オブジェクト，移動分析，マルコフ連鎖，OLAP，データキューブ

A Dynamic Histogram Construction Method to Extract Mobility Statistics from Moving Object Trajectory Streams

Yuichi TSUKAMOTO†, Yoshiharu ISHIKAWA††, and Hiroyuki KITAGAWA††

† Graduate School of Systems and Information Engineering, University of Tsukuba
†† Institute of Information Sciences and Electronics, University of Tsukuba
1–1–1 Tennoudai, Tsukuba, Ibaraki, 305-8573 Japan
E-mail: †yuichi@kde.is.tsukuba.ac.jp, ††{ishikawa,kitagawa}@is.tsukuba.ac.jp

Abstract　With the recent progress of spatial information technologies and communication technologies, it becomes easy to track trajectories of many moving objects in real-time. For the interactive analysis of huge collections of moving object trajectories, we need to accumulate given trajectory streams in an efficient and accurate manner. In this paper, we propose an algorithm to construct a dynamic histogram to summarize continual moving object trajectory streams. The histogram is based on a mobility statistics model called the Markov chain model and support an OLAP-style analysis.

Key words　moving objects, mobility analysis, markov chains, OLAP, data cube

1. まえがき

センサーや GPS 機器の小型化・低価格化に伴い，今日では移動オブジェクトの移動状況のモニタリングや集積が容易となっており，自動車の道路上での移動状況や，海洋生物の移動パターンの分析など，さまざまな**移動分析**（mobility analysis）[6] に用いられている．

多数の移動オブジェクトの移動状況を要約することは，移動オブジェクトの移動状況を蓄積した時空間データベースにおける問合せ処理でも有用である．蓄積された移動データに対し発行される問合せの選択率を効率的に推定するための研究が [2], [3] にある．

我々は，時空間データベースから移動オブジェクトの移動状況に関する統計情報を抽出する手法について研究を進めてきた．移動統計量として，**マルコフ連鎖**（Markov chain）モデルに基づく移動統計量を対象とする．時空間データ分析におけるマルコフ連鎖モデルは，ある地域から別の地域へある期間内にどの程度の人口が移動したなどの，移動オブジェクトの時空間的な移動傾向の把握に用いられる [6]．

これまで我々は，移動オブジェクトの移動軌跡が空間索引 R-木に蓄積されている状況において，R-木を用いてマルコフ連鎖の遷移確率を推定する手法の提案を行った [4], [5]．一方本研究では，ストリーム配信される移動軌跡データを継続的に集約することに焦点を当てる．

2. マルコフ過程モデルに基づく移動統計量

図 1 のように，2 次元空間が，各次元ごとに 2^P 個ずつ，$C = 2^{2P}$ 個のセルに分割されているとする（図は $P = 2$ の場合）．各セルを，以下では**空間セル**（spatial cell）と呼ぶ．ただし，曖昧さが生じない場合は単に**セル**と呼ぶ．また，図に示したような分割のことを**レベル P** の分割と呼ぶ．各セルには，$2P$ ビットのセル番号が付与されている（番号付け方式は後述）．図は，時刻 $t = \tau$ でセル 9 にいたオブジェクト A が，次の時刻 $t = \tau + 1$ でセル 12 に，そして $t = \tau + 2$ の時点でセル 6 に移動した状況を示している．

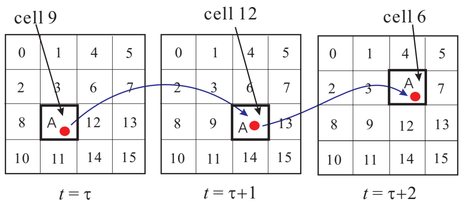

図 1　マルコフ連鎖モデルの概念

ある時刻においてセル 9 中にいて，単位時間後にセル 12 にいたオブジェクト B があったとする．B が A のようにセル 6 に進む確率（$\Pr(6|9, 12)$ と表記する）を知りたいとする．セル間の遷移を**マルコフ連鎖**（Markov chain）と考えれば，この確率は 2 次の**マルコフ遷移確率**（Markov transition probability）と捉えることができる．

上の定義を一般化する．n 次のマルコフ過程による遷移確率，すなわち，各単位時間ごとに $c_0, c_1, \ldots, c_{n-1}$ とセルを移動してきたオブジェクトが次の時点でセル c_n を訪れる確率を $\Pr(c_n|c_0, \ldots, c_{n-1})$ と表す．なお，$c_0, c_1, \ldots, c_n$（$c_i \in \{0, \ldots, C-1\}$, $0 \leqq i \leqq n$）の中には同じセルが重複していても良いとする．

データベースに多くの移動オブジェクトの移動軌跡が蓄積されているとする．移動軌跡の蓄積を開始した時刻を $t = 0$，現在の時刻を $t = T$ とする．$\Pr(c_n|c_0, \ldots, c_{n-1})$ を推定するには，各時点 $t = 0, 1, \ldots, T$ において各セルにどの移動オブジェクトが含まれているかが分かれば，その推測値は以下の式で与えられる．

$$\Pr(c_n|c_0, \ldots, c_{n-1}) = \frac{\sum_{t=0}^{T-n} |\bigcap_{i=0}^{n} \mathrm{objs}(c_i,\, t+i)|}{\sum_{t=0}^{T-n} |\bigcap_{i=0}^{n-1} \mathrm{objs}(c_i,\, t+i)|} \quad (1)$$

$\mathrm{objs}(c_i, t)$ は，時刻 t にセル c_i に含まれているオブジェクトの集合を返す関数である．ストリームとして配信される大量の移動軌跡をすべて蓄積しておき，マルコフ遷移確率の計算要求が生じた時点で式 (1) を用いて計算する方式は，単純ではあるが，蓄積されるデータ量が膨大であり，実行時のオーバヘッドが大きいという欠点がある．

本研究では，データ量を少なくし，効率的な推定処理を行うために，ヒストグラムを使用することを提案する．送られてくる移動軌跡データをセル間の遷移シーケンスとして抽出しヒストグラムに格納することで，ストリーム配信されるデータを継続的に集計し，非常にコンパクトに移動状況を表現することができる．また特徴として，ストリーム配信される遷移シーケンスに応じて動的にヒストグラムを拡張することで，必要な部分についてより詳細な遷移状況を表現することが可能となる．このヒストグラムの情報を用いてより正確な遷移確率推定を行うことができる．

3. データキューブ形式のヒストグラム表現

3.1 遷移シーケンスの抽出

ここでは，n 次のマルコフ連鎖の統計情報計算のもととなる，遷移シーケンスの抽出方式について述べる．

(id, c) というタプルのシーケンスが継続的に送られてくるとする．id はオブジェクト ID，c はセル番号であり，ID id を持つオブジェクトが，その時刻に c にいたことを表す．このようなタプルが，各単位時刻ごとに各オブジェクトごとに配信されると想定する(注1)．

$n = 2$ 次のマルコフ連鎖モデルの統計量を集計する場合，このようなシーケンスから，オブジェクトごとにセル間の遷移シーケンスを逐次検出する．たとえば，ID 1 のオブジェクトについて，$(1, 2), (1, 2), (1, 3), (1, 1), \ldots$ というタプルのストリームが配信された場合，3 個目のタプルを読んだ時点で $2 \to 2 \to 3$，4 個目のタプルを読んだ時点で $2 \to 3 \to 1$ というシーケンスを検出する．

3.2 データキューブによる遷移シーケンスの集計

遷移シーケンスの頻度を集計したものを**ヒストグラム**（histogram）と呼ぶ．本研究では，その論理表現としてデータキューブを用いる．n 次のマルコフ連鎖の場合，ヒストグラムを $n+1$ 次元のデータキューブとして構成する．$n = 2$ の場合のデータキューブ表現を図 2 に示す．これはレベル $P = 1$ の分割の場合であり，2 次元平面を $C = 2^{2P} = 4$ 個のセルに分割しているため，データキューブには $C^{n+1} = 64$ 個のセルが存在する．なお，データキューブのセルのことを以下では**キューブセル**（cube cell）と呼ぶ（曖昧さがない場合は単に**セル**と呼ぶ）．Step 0, 1, 2 という各次元は，それぞれマルコフ連鎖のステップに相当する．前節で述べた遷移シーケンス生成処理により，たとえば $1 \to 1 \to 2$ という遷移シーケンスが生成されると，対応するキューブセルに 1 が加算される．“Sum” の列は，各次元の値ごとの和を表す．

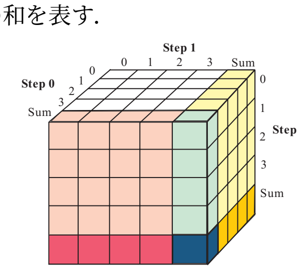

図 2　データキューブ形式のヒストグラム表現

3.3 データキューブを利用した問合せ処理

データキューブを用いると，$1 \to 1$ と遷移したオブジェクトが次にセル 2 に移動する確率は，$\Pr(2|1, 1) = val(1, 1, 2)/val(1, 1, *)$ と計算できる．ただし，$val(1, 1, 2)$ はキューブセル $(1, 1, 2)$ の値を表し，$val(1, 1, *) =$

(注1)：実際には必ずしも単位時間ごとにデータが得られない場合もある．そのような場合，内挿処理を行い単位時間ごとのシーケンスに加工する．

$\sum_{i=0}^{2^P-1} val(1, 1, i)$ の意味である．

n 次の遷移シーケンスを集計すれば，$n-1$ 次以下の遷移確率も計算できる．たとえば，セル 1 にいたオブジェクトが次にセル 2 に移る確率は，$\Pr(2|1) = val(1, 2, *)/val(1, *, *)$ となる．また，以下のような問合せも処理可能となる．

- ある時刻 $t = \tau$ にセル 1 に，$t = \tau + 2$ にセル 3 にいたオブジェクトが，$t = \tau + 1$ でセル 2 にいた確率：$val(1, 2, 3)/val(1, *, 3)$
- ある時刻 $t = \tau + 1$ にセル 2 に，$t = \tau + 2$ にセル 3 にいたオブジェクトが，$t = \tau$ でセル 1 にいた確率：$val(1, 2, 3)/val(*, 2, 3)$

3.4 ロールアップ/ドリルダウン処理の実現

OLAP でしばしば必要となるロールアップ/ドリルダウンの実現手法を，図 3 で説明する．この例では 1 次のマルコフ遷移を考えていおり，左がレベル 1 の空間分割，右がレベル 2 の空間分割である．レベル 1 のデータキューブは，レベル 2 のデータキューブを 1 段階ロールアップしたものに相当する．

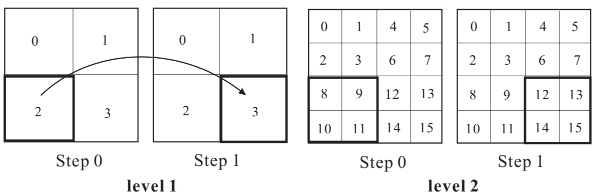

図 3　ロールアップ処理の例

下位のデータキューブと上位のデータキューブの関連について説明する(注2)．例として，図 3 のレベル 2 のデータキューブを用いて，レベル 1 の確率 $Pr(3^{(1)}|2^{(1)})$ を求める（上付き文字はレベル値である）．まず，レベル 1 のセル $2^{(1)}, 3^{(1)}$ にレベル 2 のどのセルが対応するかを見る．セル $2^{(1)}$ については，その 2 進表現が “10” であるため，レベル 2 のセルでセル番号が “10**” のパターンである，$S_0 = \{8^{(2)}, 9^{(2)}, 10^{(2)}, 11^{(2)}\}$ が対応する．同様に，セル $3^{(1)}$ については，$S_1 = \{12^{(2)}, 13^{(2)}, 14^{(2)}, 15^{(2)}\}$ が対応する．これより，

$$\Pr(3^{(1)}|2^{(1)}) = \sum_{c_0 \in S_0} \sum_{c_1 \in S_1} \Pr(c_1|c_0)$$

で計算できる．ここではロールアップの例を示したが，レベル 1 で統計量を計算していたユーザが，詳細化のためレベル 2 に移行することはドリルダウンに相当する．また，異なるレベルのセル間の遷移確率，たとえば $\Pr(10^{(2)}|2^{(1)})$ なども容易に計算できる．

4. ヒストグラムの実現方式

データキューブ形式のヒストグラムの直接的な実装はオーバヘッドが大きい．2 次元平面を各次元ごとに 2^P 個に分割した場合，空間セルの総数は $C = 2^{2P}$ 個となり，n 次のマルコフ遷移の場合には C^{n+1} 個のキューブセルが必要となる．たとえば，$P = 5, n = 2$ の場合，$C^{n+1} = 1024^3$ となり，約 10 億個のキューブセル数となり，各セル値を 2 バイトで表しても 2GB のサイズとなる．そこで，大規模なデータキューブを近似的に表現する実装方式を以下に述べる．

(注2)：ここでの論理的なデータキューブの表現に基づく説明はあくまでも概念的なもので，具体的な実装方式とは異なる．詳しくは後述する．

4.1 ヒストグラムの構築・管理手法

実装方式として，本稿では**四分木**（quadtree）に似た木構造を用いる手法を提案する．本手法は，指定されたヒストグラムのバケット数の上限値 K という制約のもとで近似的に頻度情報を表現する．なお，簡単化のため，$K = 4(4^n + m)$ という形式であるとする（m はユーザ指定の自然数とする）．詳しくは後述するが、バケット数の総数 k は，初期割り当てのバケット数 4^{n+1} と追加割り当てのバケット数 $4m$ の和となる．

4.1.1 初期状態

初期状態のデータ構造を説明する．n 次のマルコフ連鎖の場合，各ステップに対応する $n+1$ 個の空間分割を扱うが，初期状態では，各ステップに対する空間を 4 分割（レベル 1 の分割）する．初期状態では，この分割のもとで考えうるすべての n 次の遷移シーケンスに対してバケットを作成する．その総数は 4^{n+1} となる．$n = 2$ 次のマルコフ連鎖の場合，$0^{(1)} \to 0^{(1)} \to 0^{(1)}, 0^{(1)} \to 0^{(1)} \to 1^{(1)}, \ldots, 3^{(1)} \to 3^{(1)} \to 3^{(1)}$ という 64 個の遷移シーケンスのそれぞれに対しバケットを作成する．

4.1.2 成長フェーズ

次いで，構築された 4^{n+1} 個のバケットからなるヒストグラムを初期状態として遷移シーケンスの集計を開始する．集計が進むにつれカウント数のばらつきが広がるため，カウント数が閾値 θ を超えたバケットを細分化する．後述のように，1 回の細分化処理で新たなバケットが 4 個追加されるため，m 回の細分化で $K = 4(4^n + m)$ に達する．バケット数が K に至るまでの，初期的なヒストグラム構造を構築する段階を成長フェーズと呼ぶ．

$n = 2$ 次のマルコフ連鎖に対するヒストグラムの例を用いて，細分化処理を説明する．初期状態のヒストグラムに対し集計を進めたところ，$0^{(1)} \to 1^{(1)} \to 2^{(1)}$ という遷移シーケンスに対するバケットのカウント値（$\#(0^{(1)} \to 1^{(1)} \to 2^{(1)})$ で表す）が閾値 θ に達したとする．ここでこのバケットを 4 つに細分化するが，以下の $n + 1 = 3$ 通りから最良のものを選択する（選択方式については後述）．

（1） ステップ 2 を細分化する場合：$0^{(1)} \to 1^{(1)} \to 8^{(2)}, 0^{(1)} \to 1^{(1)} \to 9^{(2)}, 0^{(1)} \to 1^{(1)} \to 10^{(2)}, 0^{(1)} \to 1^{(1)} \to 11^{(2)}$ の 4 つの遷移シーケンスに対するバケットを作成する．

（2） ステップ 1 を細分化する場合:$0^{(1)} \to \{4^{(2)}, 5^{(2)}, 6^{(2)}, 7^{(2)}\} \to 2^{(1)}$ に対する 4 つのバケットを作成する．

（3） ステップ 0 を細分化する場合:$\{0^{(2)}, 1^{(2)}, 2^{(2)}, 3^{(2)}\} \to 1^{(1)} \to 2^{(1)}$ に対する 4 つのバケットを作成する．

作成した 4 つの各バケットのカウント値には初期値 $\theta/4$ を設定する．

細分化処理をブロック数が K になるまで行うことで，図 4 のようなような木構造が作られる．ルートノードは，初期時点で作られた 4^{n+1} 個のバケットを表す．ルートノードの各エントリには，バケットのカウント数と，4 つのバケットを含む子ノードへのポインタが含まれる．子ノードへの矢印に表示されている “div = step 1” などの文字列は，マルコフ連鎖のどのステップについて細分化したかを示している．

4.1.3 継続処理フェーズ

$K = 4(4^n + m)$ 個のバケットが割り当てられると継続処理

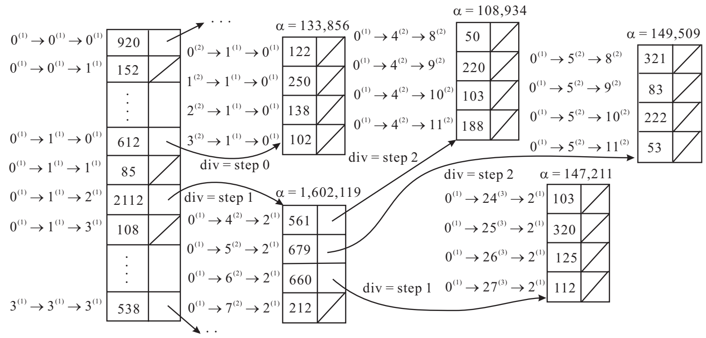

図 4　ヒストグラムの物理構造

のフェーズに入る．遷移シーケンスの集計を続けるうちに，バケットのカウント値のばらつきが広がっていくため，ヒストグラムを再構成し，バケット数が一定という制約のもとで，より効率的な構成を実現する．

K 個のバケットのうち，細分化されていないバケット（ポインタが出ていないバケット）は $L = 4^{n+1} + 3m$ 個ある．これらを $B_1, \ldots, B_L$ とする．バケットの分割の良さを示すため，以下の指標を導入する．

$$BV = (\textstyle\sum_{i=1}^{L}(v_i - \bar{v})^2)/\bar{v} \qquad (2)$$

ただし，v_i はバケット B_i のカウント数であり，

$$\bar{v} = (\textstyle\sum_{i=1}^{L} v_i)/L \qquad (3)$$

である．すなわち，BV はバケットのカウント値の分散である．

次に，ヒストグラムの再構成方式を述べる．バケット B_p $(1 \leqq p \leqq L)$ のカウント数が多いため，これを細分化したいとする．バケット数の総数を L に保つため，細分化により生成された子ノードのうち，リーフレベルのものを 1 つ選択して削除する（子ノードを持っているバケットを削除できないため）．たとえば，図 4 には，表示されているものだけでは 5 つの子ノードが存在するが，このうちリーフレベルにある 4 つが削除の候補となる．

削除対象の候補のある子ノードに含まれる 4 つのバケットを $B_{q,0}, B_{q,1}, B_{q,2}, B_{q,3}$ とすると，以下の性質が成り立つ（証明略）．

［性質 1］　B_p を 4 つに細分化し，その代わりに 4 つのバケット $B_{q,0}, B_{q,1}, B_{q,2}, B_{q,3}$ を削除しても，カウント値の平均 $\bar{v}$ は不変である．また，処理後のカウント値の分散の値を BV' とすると，

$$BV - BV' = \frac{3v_p^2 - 8(\sum_{i=0}^{3}\sum_{j=i+1}^{3} v_{q,i}v_{q,j})}{4\bar{v}} \qquad (4)$$

という関係が成立する．　□

よって，$BV - BV' > 0$ の場合，再構成を行えば分散を削減できる．特に，v_p の値が最大である B_p と，$\sum_{i=0}^{3}\sum_{j=i+1}^{3} v_{q,i}v_{q,j}$ の値が最小であるバケットの組 $B_{q,0}, B_{q,1}, B_{q,2}, B_{q,3}$ を選ぶことで，分散を最も減らすことができる．そこで，以下の再構成方式を用いる．

ヒストグラムの再構成方式：最大の v_p を持つバケット B_p と，リーフレベルのある子ノードで，$\sum_{i=0}^{3}\sum_{j=i+1}^{3} v_{q,i}v_{q,j}$ が最小なバケットの組 $B_{q,0}, B_{q,1}, B_{q,2}, B_{q,3}$ について，$BV - BV' \geqq \mu$ の場合に再構成処理を行う．

微小な再構成が頻繁に生じることを避けるため，閾値 μ を設ける．

再構成の対象となるバケットを高速に特定するため，図 4 のように，各子ノードに対する α 値を維持管理する．そのノードに含まれる 4 つのバケットを $B_{r,0}, B_{r,1}, B_{r,2}, B_{r,3}$ としたとき，α 値を

$$\alpha = \textstyle\sum_{i=0}^{3}\sum_{j=i+1}^{3} v_{r,i}v_{r,j} \qquad (5)$$

で定義する．α 値の管理は容易である．新たな遷移シーケンスの到着によりあるバケット $B_{r,i}$ $(0 \leqq i \leqq 3)$ をインクリメントする際，

$$\alpha \leftarrow \alpha + \textstyle\sum_{j=0,\, j \neq i}^{3} v_{r,j} \qquad (6)$$

で対応するノードの α 値を更新すればよい．

更新処理の概略を述べる．木構造以外に，(1) バケットのカウント数の平均値 $\bar{v}$，(2) 最大のカウント数のバケット，(3) 最小の α 値を持つ子ノード，の情報を維持管理する．新たな遷移シーケンスが配信されると，ルートノードから対応するバケットが見つかるまで木構造をたどり，カウント値をインクリメントする．途中で出会ったバケットについてもインクリメントを行う．子ノードについては同時に α 値も更新する．カウントの更新処理が終わると，最大のカウント値を持つバケットと最小の α 値を持つ子ノードの情報を更新し，

$$\bar{v} \leftarrow (L\bar{v} + 1)/L \qquad (7)$$

により値を更新し，式 (4) により再構成を行うかどうかを判定する．具体的な分散の値を計算しなくてよいため，効率的な処理となる．

4.2　細分化処理の選択方式

4.1.2 で，細分化の際に $n+1$ 通りの候補から最良の 1 つを選ぶと述べた．以下では，遷移シーケンス $0^{(1)} \to 1^{(1)} \to 2^{(1)}$ に対するバケットの細分化を例として説明する．$0^{(1)} \to 1^{(1)} \to 2^{(1)}$ のバケットのカウント数が非常に多いという状況を以下のように分析できる．

- $\Pr(2^{(1)}|0^{(1)}, 1^{(1)}) = \#(0^{(1)} \to 1^{(1)} \to 2^{(1)})/(\sum_{i=0}^{3}\#(0^{(1)} \to 1^{(1)} \to i^{(1)}))$ が大きい場合：たとえば 100% に近いときは，$0^{(1)} \to 1^{(1)}$ と移動したオブジェクトがでほとんど $2^{(1)}$ に移動することから，ステップ 2 を細分化するメリットが大きいと考えられる．
- 一方，$\#(0^{(1)} \to 1^{(1)} \to 2^{(1)})/(\sum_{i=0}^{3}\#(0^{(1)} \to i^{(1)} \to 2^{(1)}))$ の値が大きい場合，ステップ 1 を細分化するのがよいと考えられる．
- 同様に，$\#(0^{(1)} \to 1^{(1)} \to 2^{(1)})/(\sum_{i=0}^{3}\#(i^{(1)} \to 1^{(1)} \to 2^{(1)}))$ の値が大きい場合，ステップ 0 の細分化が望ましいといえる．

以上の考察に基づいて，上に挙げた $n+1 = 3$ 個の値を比較し

て，最大の値となるステップについて分割を行う．

上の例は最も単純な場合であるが，たとえば $0^{(1)} \to 6^{(2)} \to 2^{(1)}$ を細分化する場合には，

- $\#(0^{(1)} \to 6^{(2)} \to 2^{(1)})/(\sum_{i=0}^{3} \#(0^{(1)} \to 6^{(2)} \to i^{(1)}))$
- $\#(0^{(1)} \to 6^{(2)} \to 2^{(1)})/(\sum_{i=0}^{3} \#(0^{(1)} \to i^{(1)} \to 2^{(1)}))$
- $\#(0^{(1)} \to 6^{(2)} \to 2^{(1)})/(\sum_{i=0}^{3} \#(i^{(1)} \to 6^{(2)} \to 2^{(1)}))$

の 3 つの値を比較することになる．

4.3 遷移確率の推定処理

マルコフ遷移確率の推定処理は，木構造で表現されたヒストグラムから遷移シーケンスの出現回数を推定し集計する処理に帰着できる．よって，ここでは遷移シーケンスの出現回数の推定アルゴリズムの概略を述べる．アルゴリズムを図 5 に示す．

関数 $count$ の処理を図 4 を例にして説明する．$0^{(1)} \to 1^{(1)} \to 1^{(1)}$ の推定は，対応するバケットがルートノードに存在するため，3 行目で終了し 85 が返る．これに対し，$0^{(1)} \to 6^{(2)} \to 2^{(1)}$ や $0^{(1)} \to 1^{(1)} \to 8^{(2)}$ については，より正確な推定値が $0^{(1)} \to 1^{(1)} \to 2^{(1)}$ のポインタの先の子孫ノードより計算できる可能性があるため，$child\_count$ 関数を呼出しする．一方，$0^{(1)} \to 6^{(2)} \to 1^{(1)}$ や $0^{(1)} \to 19^{(3)} \to 5^{(2)}$ については，該当するバケットが存在するなら $0^{(1)} \to 1^{(1)} \to 1^{(1)}$ の子孫であるのだが，該当する子孫ノードは存在しない．この場合，7 行目の関数 $estimate$ で近似値を推定する．

関数 $estimate$ の推定手法を述べる．$n = 2$ 次のヒストグラム上での $n = 2$ 次の出現回数の推定について説明する．出現回数を推定する遷移シーケンスを $i^{(p)} \to j^{(q)} \to k^{(r)}$ とし，推定に用いるバケットの遷移シーケンスを $i'^{(p')} \to j'^{(q')} \to k'^{(r')}$ とする．このとき，

$$\#(i^{(p)} \to j^{(q)} \to k^{(r)}) = \frac{\#(i'^{(p')} \to j'^{(q')} \to k'^{(r')})}{4^{(p'-p)+(q'-q)+(r'-r)}} \quad (8)$$

で出現回数を推定する．たとえば図 4 では，$0^{(1)} \to 6^{(2)} \to 1^{(1)}$ については $85/4$，$0^{(1)} \to 19^{(3)} \to 5^{(2)}$ については $85/4^3$ と見積もる．n の値が異なる場合の処理は同様に行える．

次に，$child\_count$ 関数が呼び出される場合について説明する．図 4 で $0^{(1)} \to 6^{(2)} \to 2^{(1)}$ の出現回数を推定する場合，子ノードをたどると対応するバケットにアクセスできるため，2 行目で終了し 660 を返す．一方，$0^{(1)} \to 5^{(2)} \to 8^{(2)}$ に対し出現回数を推定する際は，ルートノードの $0^{(1)} \to 1^{(1)} \to 2^{(1)}$ のポインタをたどった子ノードの中から適切なバケットを選択する．この場合，$0^{(1)} \to 5^{(2)} \to 8^{(2)}$ に対する子ノードをその先に含みうるのは $0^{(1)} \to 5^{(2)} \to 2^{(1)}$ のバケットである（このような関係を $0^{(1)} \to 5^{(2)} \to 2^{(1)} > 0^{(1)} \to 5^{(2)} \to 8^{(2)}$ という半順序で示す）．よって，$0^{(1)} \to 5^{(2)} \to 2^{(1)}$ のバケットにポインタがあれば，5 行目で先にたどることになる．これとは異なり，$0^{(1)} \to 7^{(2)} \to 8^{(2)}$ に対する推定の場合は，対応する $0^{(1)} \to 7^{(2)} \to 2^{(1)}$ のバケットにポインタがないため，8 行目で近似値を $212/4$ と推定する．

$0^{(1)} \to 1^{(1)} \to 8^{(2)}$ の場合には，以上の場合に当てはまらないため，10〜18 行目で処理する．まず，$divide\_sequence(0^{(1)} \to 1^{(1)} \to 8^{(2)}, 1)$ により，遷移シーケンス $0^{(1)} \to 1^{(1)} \to 8^{(2)}$ を $0^{(1)} \to 4^{(2)} \to 8^{(2)}, \ldots 0^{(1)} \to 7^{(2)} \to 8^{(2)}$ の 4 つのシーケンスに分解する．この関数は，第 2 引数で与えられたステップについて，与えられた遷移シーケンスを分解した配列を返す関数である．分解したそれぞれの遷移シーケンスについて，11〜17 行で出現回数の推定を行い，18 行目で集計結果を返す．

```
function count(qseq)
1 ルートノードで qseq に対応するバケット b を特定する;
2 if b.seq ≡ qseq then // シーケンスが一致するバケットが存在
3    return b.count; // カウント値を返し終了
4 if b.ptr ≢ NULL then // 子ノードが存在
5    return child_count(b.ptr, qseq); // 子孫ノードからの推定値を返す
6 // 出現回数の推定値を計算し返す
7 return estimate(node.b.count, node.b.seq, qseq);
function child_count(node, qseq)
1 if node.bucket[i].seq ≡ qseq である i (0 ≦ i ≦ 3) が存在 then
2    return node.bucket[i].count; // カウント値を返し終了
3 if node.bucket[i].seq > qseq である i (0 ≦ i ≦ 3) が存在 then
4    if node.bucket[i].ptr ≢ NULL then // 子ノードが存在
5       return child_count(node.bucket[i].ptr, qseq);
7    end
8    return estimate(node.bucket[i].count, node.bucket[i].seq, qseq);
9 end
10 qseqs := divide_sequence(qseq, div);
11 for i := 0 to 3 do
12    if node.bucket[i].ptr ≢ NULL then
13       cnt[i] := child_count(node.bucket[i].ptr, qseqs[i]);
14    else
15       cnt[i] := estimate(node.bucket[i].count, node.bucket[i].seq, qseqs[i]);
16    end
17 end
18 return Σ_{i=0}^{3} cnt[i];
```

図 5 出現回数の推定アルゴリズム

4.4 コストの見積もり

遷移シーケンスの出現確率の推定の場合には，ルートノードからバケットの探索を行う．n 次のヒストグラム上で n 次の遷移シーケンスの出現回数を推定するときには，木構造が最悪の場合，m 個の子ノードをたどることから，最少で 1 個，最多で $m+1$ 個のバケットへのアクセスとなる．よって $O(m)$ の計算量となるが，基本的には単純な木構造でポインタをたどる処理に帰着される．マルコフ遷移確率の推定の場合には，4 個の遷移シーケンスの出現回数の推定を行い，それをもとに計算を行うため，約 4 倍の計算コストとなる．

次に，ヒストグラムの定常的な維持管理のための計算コストについて述べる．遷移シーケンスが配信されると，該当バケットに達するまで木構造をたどるため，$O(m)$ の処理となる．一方，リストなどを管理する処理は比較的軽微と考えられる．

細分化処理が発生した場合には，$n+1$ 個の細分化手法のそれぞれについて，4 個の遷移シーケンスの出現回数の推定値を算出する必要があり，$O(nm)$ の処理となる．ただし，n の値は小さいため，実際には $O(m)$ であり，出現回数の推定処理の定数倍の処理時間ですむと考えられる．

次に，データ構造の記憶コストを考える．カウント数とポインタを 4 バイトで表すと，木構造のサイズは $8 \times 4(4^n + m) \approx 32m$ バイトとなる．2 つのリストも $O(m)$ のサイズであり，全体の記憶コストは $O(m)$ となり，$m = 1{,}000$ 程度であれば数十〜数百 KB と想像される．よって，バケットの総数 K の設定に

依存するが，データキューブをそのまま実現するのに比べ，非常にコンパクトなデータ構造となり，メモリ上に常時置くことにまったく問題はないと考えられる．

5．実　　験

この章では，実際の移動軌跡データを用いて実験を行い，提案するヒストグラム構築手法の有効性を示す．主に２つの項目について評価を行う．

（1） 処理時間：

- ヒストグラム構築時間：ヒストグラム構築時間に関連する値としては，入力データ数・バケット分割閾値 θ・最大バケット数 K がある．これらの値を変化させたときの構築時間の変化を比較する．
- 遷移確率推定処理時間：推定処理時間は，ヒストグラムの大きさに関連するため，異なる大きさのヒストグラムに対する，推定問合せ１つあたりの実行時間を比較する．

（2） 正確さ：ヒストグラムから求めた遷移確率がどのくらい正確であるかを，実際のデータから得た遷移確率と比較することで評価を行う．

5.1 実験用データ

実験に用いるデータは，データ生成ソフトウェア [1] を用いて作られた移動軌跡データ 100 万件を対象とする．

5.2 実 験 結 果

典型的な例としては，100 万件の移動軌跡データを，最大バケット数 $K = 4064$，バケット分割閾値 $\theta = 100$，再構成閾値 $\mu = 10$ とした場合，木構造は 9 のレベルまで，遷移シーケンスとしては最大 $0^{(4)} \rightarrow 0^{(5)} \rightarrow 0^{(3)}$ まで分割されたヒストグラムが構築された．

5.2.1 処 理 時 間

a） ヒストグラム構築時間

配信される移動軌跡データを追従できる程度のヒストグラム構築時間が可能かどうかを調べる．

配信されるデータを 10 万件から 100 万件まで 10 万件刻みで用意したときの，ヒストグラム構築時間を測定する．

最大バケット数 K を 4064 としたときのヒストグラム構築時間を図 6 に示す．4 本のグラフはそれぞれ θ を変化させたものである．図中の矢印は成長フェーズから継続処理フェーズに切り替わる瞬間を示している．

1 データをヒストグラムに挿入するのにかかる平均時間は，成長フェーズ時には 0.05 ミリ秒，継続処理フェーズ時には，0.99 ミリ秒となる．継続処理フェーズには，再構成を伴う挿入と伴わない挿入の処理があるが，再構成を伴う場合には 1.8 ミリ秒，伴わない場合には 0.97 秒かかる．なお，最大バケット数 $K = 4064$，バケット分割閾値 $\theta = 100$，再構成閾値 $\mu = 10$ としたときには，継続処理フェーズ 887875 回中再構成が 151 回発生した．継続処理フェーズに入ると処理に時間がかかるため，構築時間が大きくなることがグラフからわかる．$\theta = 1000$ のときは，一度も継続処理フェーズに入らなかったため，100 万件を約 50 秒という速度で構築することができた．

次に，先ほどと同様に最大バケット数 6064 件と 8064 件で構築した場合の図を 7,8 に示す．

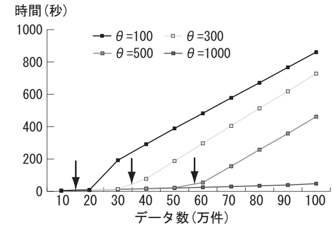

図 6　最大バケット数 4064 個のときのヒストグラム構築時間

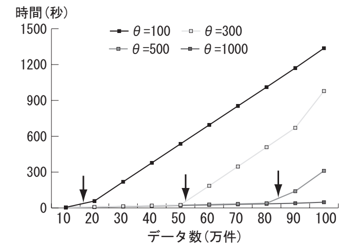

図 7　最大バケット数 6064 個のときのヒストグラム構築時間

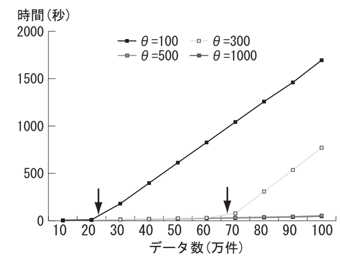

図 8　最大バケット数 8064 個のときのヒストグラム構築時間

速度変化の傾向は先ほどと同様である．θ の値が小さいと，すぐにバケット分割が起こってしまうため継続処理フェーズに入るまでの時間が短くなっていることがわかる．この継続処理フェーズに入るまでの時間は，最大バケット数を増加させることで長くすることが出来る．ただし，一旦継続処理フェーズに入ってしまうと，4.1.3 節で述べたように，バケットの情報をすべて保持しなければならないため処理に時間がかかる．実際には，1 データ挿入にかかる時間は，$K = 6064$ のときに成長フェーズで 0.05 ミリ秒，継続処理フェーズで 1.59 ミリ秒となる．再構成を伴う場合は 2.9 ミリ秒，伴わない場合は 1.4 ミリ秒となり，継続処理フェーズ 830081 回中 250 回の再構成が発生した．$K = 8064$ のときには，成長フェーズに 0.06 ミリ秒，継続処理フェーズに 2.08 ミリ秒となる．再構成を伴う場合には 4.12 ミリ秒，伴わない場合は 1.9 ミリ秒となり，継続処理フェーズ 777217 回中 341 回の再構成が発生した．

また，図からわかるように，入力されるデータ数が増加しても，継続処理フェーズにおけるデータ 1 件あたりの挿入コストはほぼ定数でヒストグラムを構築することが出来る．

b） 遷移確率推定時間

空間分割レベルを 1,2,3 として遷移確率を推定したときの，遷移シーケンス１つあたりの推定時間を比較する．分割レベルを 1 とした場合，空間は (2×2) に分割され，2 次の遷移確率を求めるため，$0^{(1)} \rightarrow 0^{(1)} \rightarrow 0^{(1)} \sim 3^{(1)} \rightarrow 3^{(1)} \rightarrow 3^{(1)}$ の全部で 64 通りの遷移シーケンスの組み合わせが考えられる．同様にレベル 2 の場合には，空間を (4×4) に分割するのでシーケンスの組み合わせは $0^{(2)} \rightarrow 0^{(2)} \rightarrow 0^{(2)} \sim 15^{(2)} \rightarrow 15^{(2)} \rightarrow 15^{(2)}$ の

4096 通り，レベル 3 の場合には 262144 通りになる．これらのシーケンスに対する問合せにかかる時間を計測し，1 問合せあたりの平均時間を比較した．

問合せ速度は θ の値には関連せず最大バケット数 K に関連するため，$\theta = 100$ に固定したときの 1 問合せにかかる時間を図 9 に示す．

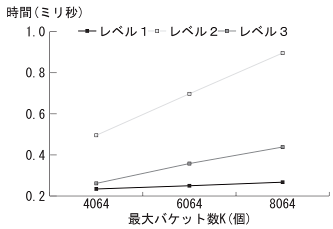

図 9　1 問合せ実行にかかる時間

最大バケット数が多くなると，それに応じて問合せ時にアクセスするバケット数も増加するため，1 問合せ実行にかかる時間も増加する．

レベル 2 の遷移シーケンスを求めるときの方がレベル 3 のときよりも問合せ実行時間が長いのは，レベル 2 のシーケンスを示すバケットは比較的ルートノードに近いところにあるが，一致しない場合にはその下のレベルのノードもアクセスして予測値を計算しなければならないためである．対象とするバケットの下のレベルに存在するバケット数はレベル 3 よりもレベル 2 のバケットを対象としたときの方が多いため，予測値を求める計算に時間を要していると考えられる．

しかし，最大でも 1 ミリ秒に満たない計算時間しかかからないため，ストリーム配信されるデータをヒストグラムに蓄積しながら問合せを実行しても問題ないと考える．

5.2.2　ヒストグラムによる遷移確率推定の正確さ

a）ユークリッド距離による比較

ヒストグラムの正確さを測定するために，ユークリッド距離を使う手法を用いる．

この方式では，空間分割レベル P(2^{2P} 個のセルに分割) とし，k 次のマルコフ過程を考えたとき，それぞれの遷移シーケンス seq に対する頻度の推定値を $est(seq)$ とし，正確な値を $real(seq)$ とすると，

$$dist = \sqrt{\sum_{i=1}^{(2^{2P})^k} |est(seq_i) - real(seq_i)|^2} \tag{9}$$

という式で距離を計算する．この値が小さいほどヒストグラムの精度が高いといえる．

例えば，空間をレベル 1 の空間分割 (2×2) にしたときの 2 次のマルコフ過程を考えると，$0^{(1)} \to 0^{(1)} \to 0^{(1)}, \ldots, 3^{(1)} \to 3^{(1)} \to 3^{(1)}$ という 64 通りの組み合わせが考えられる．よって

$$dist = \sqrt{\sum_{i=1}^{64} |est(seq_i) - real(seq_i)|^2} \tag{10}$$

という式になる．

このユークリッド距離を，以下の値の設定のもとで比較を行う．

- 空間分割レベル：レベル 1=2×2 に空間を分割，レベル 2=4×4 に空間を分割，レベル 3=8×8 に空間を分割の 3 種類を比較
- バケット分割閾値 θ：100，300，500，1000 の 4 種類
- 最大バケット数 K：$K = 4064(m{=}1000)$，$6064(m{=}1500)$，$8064(m{=}2000)$ の 3 種類

b）評価結果

空間分割レベル $1(2 \times 2)$ における問合せの考えられる遷移シーケンスの組み合わせは 64 通りであるが，これはヒストグラムのルートノードに当たる．ヒストグラムのノードは分割されていないため，常に正確な値を返すことが出来る．

次に，空間分割レベル $2(4 \times 4)$ における問合せ (4096 通りのシーケンス) の結果を図 10 に示す．横軸に分割閾値 θ をとったときの 3 種類のヒストグラムの推定精度を表す．$\theta = 100$ で分割したヒストグラムは，最大バケット数が増加するにつれて，精度が高くなる．これは，バケット数が多い方がより詳細に移動オブジェクトのシーケンスを蓄積できるためであると考えられる．それ以外の θ の値では，あまり差が見られなかったが，全体的に高い精度を示している．

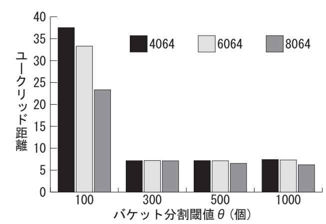

図 10　レベル 2 の空間分割における遷移確率推定の精度

次に，空間分割レベル $3(8 \times 8)$ における問合せ (262144 通りのシーケンス) の結果を図 11 に示す．$\theta = 100$ の時は，レベル 2 のときよりも精度が良いという結果が出ているが，全体的にレベル 2 の分割のときよりも，精度が落ちている．この原因としては，レベル 2 のときに比べて，シーケンスが細かくなるため問合せシーケンスと一致するバケットが少なくなると考えられる．その結果，確率推定時に用いる値に，関数 estimate() から計算された近似値の比率が高くなってしまうためである．

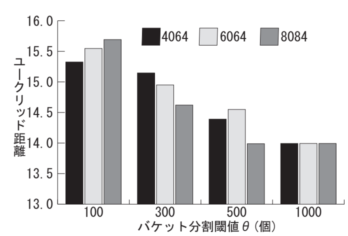

図 11　レベル 3 の空間分割における遷移確率推定の精度

c）考　　察

バケット分割閾値 θ の値を小さくすると精度が落ちる．これは，バケットのカウント数が小さいうちから分割してしまうため，バケットの構造が実際の確率分布にうまくマッチしないことが多いためである．そのため，細かく分割したシーケンスと問合せシーケンスが一致した場合には正確な値を返すことが出

来るが，そうでない場合には誤差が大きくなってしまう．これに対して θ の値を大きくすると，estimate() の誤差が小さくなる．問合せ全体の処理を考えると，estimate() を呼び出す回数は非常に多いため，θ を小さくすると精度が落ちるといえる．

また，バケット数が増加しても精度が上がらない原因としては，バケット分割が局所的に細かくなってしまうためである．本実験では比較的粗い分割のもとで精度を評価しているため，細分化したメリットがあまり出ていないと考えられる．

全体的な精度に関しては，一番のネックは，バケット分割時にカウント値を均等に 4 等分するという点である．これによって 2 つの問題が出てくる．1 つは，オブジェクトの移動状況を適切に表現できないという問題である．例えば，$1^{(2)} \to 1^{(2)} \to 0^{(1)}$ というバケットをステップ 2 について細分化する場合，$1^{(2)} \to 1^{(2)} \to 0^{(2)}, \ldots, 1^{(2)} \to 1^{(2)} \to 3^{(2)}$ という 4 つのシーケンスに分割することになる．このとき，$1^{(2)} \to 1^{(2)}$ と遷移したオブジェクトは次のステップで $1^{(2)}$ に遷移する可能性が一番高くなり，実際のカウント値には偏りがあることは明らかであるが，このような移動状況を表すことができない．ただし，十分に小さいセルに分割したときには遷移確率は 4 つとも同じであるとみなすことができるため，誤差は無視することができる．2 つ目は，確率 0.0 のシーケンスを正確に検出できないという点である．例えば，$0^{(2)} \to 5^{(2)} \to 15^{(2)}$ というシーケンスは一般には見られないが，$3^{(2)} \to 6^{(2)} \to 12^{(2)}$ というシーケンスは多いという場合を考える．この 2 つのシーケンスは，ルートノードレベルで考えると，どちらも $0^{(1)} \to 1^{(1)} \to 3^{(1)}$ となる．提案手法ではバケット分割時にカウント値も均等に分けるため，実際にありえないシーケンスについてもカウントしてしまう．そのため，全体的に見るとカウント 0 のシーケンスというものがほとんどなくなってしまう．実際にレベル 2 分割の場合 4096 通りの遷移シーケンスが考えられるが，正確には 3645 個のシーケンスは確率 0.0 であった．しかし提案手法では 1066 個のシーケンスしか確率 0.0 を検出できなかった．これらの問題が全体的な精度を落としている原因だと思われる．

よって，精度を上げる方法として，ルートバケットの空間分割レベルを $2(4 \times 4)$ から開始することを考えた．ルートレベルのバケット数は 4096 個となるが，コストとしては問題ないと考える．こうすることで，空間分割レベル 1，2 の問合せシーケンスについては精度を 100%にすることができる．またレベル 1 から 2 への分割がなくなることで，分割によるカウント値の分散が小さくなり，確率 0.0 の検出精度が上がると予想される．

図 12 に，空間分割レベル $3(8 \times 8)$ の問合せにおける結果を示す．ルートバケットの空間分割レベルが 2 のヒストグラムは最大バケット数を 8096 個と 10096 個とした．

θ の値を小さくすると精度が落ちるという傾向は変わらない．ルートバケットレベルが 2 の場合には，確率 0.0 の検出精度は上がったが，バケット分割によるカウント値の分散が減る分だけ誤差が大きくなる．そのため，全体的な精度は落ちている．

実験から，粗い空間分割における遷移確率推定には，ルートバケットレベルを大きくし，細かい空間分割における遷移確率推定には，ルートバケットレベルを 1 にするのが良いと考えられる．このことから，求めたい遷移シーケンスに応じて，適切なルートバケットレベルを設定することが重要であると考えられる．

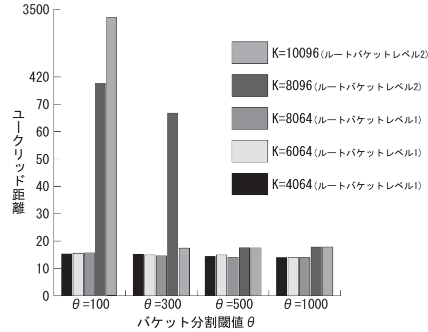

図 12　レベル 3 の空間分割における遷移確率推定の精度

6. ま と め

本稿では，時空間データベースの情報からマルコフ過程モデルに基づく移動統計量を効率よく集計するためのヒストグラム手法の提案を行った．OLAP 的な移動分析処理のアプローチについて述べ，論理的なデータキューブ構造を支援するための物理的なデータ構造の構築手法について述べた．

提案手法によるヒストグラムは，大量のデータに対しても短い時間で構築できるため，ストリーム配信される移動軌跡データを継続的に集計して格納し，遷移確率推定を行うことが出来ることを示した．

遷移確率の精度に関しては，バケット分割時に格納されている集計値を均等に分割してしまうためこの部分が精度を落としていると考えられる．より正確に集計値を分割するアルゴリズムを考える必要があると考えられる．

さらに，移動オブジェクトの集合が非定常的（nonstationary）な遷移確率に従い，時間とともに移動パターン自体が変化するような状況に，本手法がどの程度追従できるかについても検証を行いたい．移動窓や忘却パラメータの導入などによる，比較的近い過去を対象としたヒストグラムへの拡張についても検討したい．

謝辞　本研究の一部は，文部科学省科学研究費特定領域研究 (2)(15017207)，日本学術振興会科学研究費基盤研究 (B)(15300027)，若手研究 (B)(14780316) による．

文　献

[1] T. Brinkhoff. A framework for generating network based moving objects. In *GeoInfomatica*, No. 6(2), pp. 153–180, 2002.
[2] Y. Choi and C. Chung. Selectivity estimation for spatio-temporal queries to moving objects. In *Proc. ACM SIGMOD*, pp. 440–451, 2002.
[3] Y. Tao, J. Sun, and D. Papadias. Selectivity estimation for predictive spatio-temporal queries. In *Proc. ICDE*, 2003.
[4] 塚本祐一, 石川佳治, 北川博之. 索引付けされた移動軌跡データからの移動統計量の抽出法の評価. 電子情報通信学会技術研究報告, Vol. 103, No. 191, pp. 31–36, 2003.
[5] 塚本祐一, 石川佳治, 北川博之. 索引付けされた移動軌跡データからの効率的な移動統計量抽出法. 日本データベース学会 Letters, Vol. 2, No. 1, pp. 27–30, 2003.
[6] G. J. G. Upton and B. Fingleton. *Spatial Data Analysis by Example, Volume II: Categorical and Directional Data*. John Wiley & Sons, 1989.