情報処理学会論文誌　データベース　Vol. 4　No. 1　26–39 (Mar. 2011)

I-Tree：センシングデータの統合検索を支援する空間時系列索引機構

木實新一[†1,†6]　石塚宏紀[†2,†6]　岩井将行[†3]
宮崎純[†4]　瀬崎薫[†5]　戸辺義人[†1,†6]

物理空間の各所に設置されたセンサのデータを統合的に検索し，適切なタイミングでユーザに情報やサービスを提供するためには，空間時系列データを高速に処理する必要がある．本論文では，大量の空間時系列データの高速な検索を可能にする空間時系列索引機構 I-Tree の構造と検索アルゴリズムについて述べ，人工的に生成した大量の人数計測データと市街地に設置した微気象センサのデータを用いた評価実験の結果を示す．I-Tree を用いれば，空間時系列問合せに対してユークリッド距離の近い系列を，従来手法よりも安定して高速に検索することができる．

I-Tree: A Spatial Time-series Indexing Mechanism for Supporting Integrated Retrieval of Sensing Data

Shin'ichi Konomi,[†1,†6] Hiroki Ishizuka,[†2,†6]
Masayuki Iwai,[†3] Jun Miyazaki,[†4] Kaoru Sezaki[†5]
and Yoshito Tobe[†1,†6]

Fast and efficient retrieval of spatial time series data is indispensable for applications that must provide users with timely and relevant information and services using complex sensor data in the real world. This paper introduces an indexing mechanism called I-Tree, which allows for fast retrieval of a massive amount of spatial time-series data, and describes its structure and the search algorithms. We also discuss the results of our performance evaluation using a large amount of artificially-generated spatial time-series data as well as temperature data from micro-weather sensors deployed in a city. I-Tree allows for fast, Euclidean distance-based similarity search of spatial time-series under various conditions.

1. はじめに

センシング技術の発展により，地理空間や人間活動に関する詳細なデータを大量に取得することが可能となりつつある．様々な場所に設置された多種多様なセンサが生成するデータを活用して，市民や専門家に対して有用な情報を提供するためには，空間的な広がりを持つ時系列データ（空間時系列データ）を効率良く処理することが必要である．空間時系列データを効率良く処理することができれば，市民に対してタイムリに有用情報や警報情報を提供できる可能性がある．また，効率的な分析やインタラクティブな可視化によって，都市や社会システムに関する新たな知見を得たり，社会における合意形成や問題解決のプロセスを円滑化したりすることができる可能性がある．

空間時系列データに基づく能動的な情報提供によって，たとえば以下のシナリオに示すようなサービスを実現することができる．

[混雑警報のシナリオ] 東京ドームでイベントが終了した直後は，最寄りの 3 つの駅（水道橋駅，後楽園駅，春日駅）で混雑が発生する．そこで，周辺の地域にセンサ（ステレオカメラ，レーザレンジスキャナ等）を設置し，要所を通過する人数を計測する．市民のプライバシを保護するために，人数のみを計測することとし，個々の歩行者の識別や追跡は行わない．各所に設置したセンサが発する時系列をすべてまとめた空間時系列を常時監視しつつ，過去の空間時系列はデータベースに蓄積する．最近 3 分間の空間時系列と似た系列を定期的にデータベースで検索し，検索によって得られた系列の時刻から数分以内に水道橋駅で混雑が発生した記録がある場合，現在水道橋駅の近くにいる人々に対して混雑警報を発信する．さらに，同地域に微気象センサを設置すれば，混雑度と気温，湿度を総合的に考慮して

†1 東京電機大学未来科学部
School of Science and Technology for Future Life, Tokyo Denki University
†2 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, the University of Tokyo
†3 東京大学生産技術研究所
Institute of Industrial Science, the University of Tokyo
†4 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
†5 東京大学空間情報科学研究センター
Center for Spatial Information Science, the University of Tokyo
†6 科学技術振興機構戦略的創造研究推進事業
JST-CREST

人々に快適な歩行経路を推薦することも可能である．

ある時刻における空間的な人数分布だけを見て人々の移動方向を知ることは難しいが，人数分布の時間変化を見れば，集団の移動方向を検出し混雑を予測することが容易になる．空間時系列を検索すれば人数分布の時間変化を知ることができるが，混雑が発生する前に警報を発信するためには，大量の空間時系列を非常に効率良く処理する必要がある．

本論文では，大量の空間時系列の高速な検索を可能にする索引機構 I-Tree[26),27)] の構造と検索アルゴリズムについて述べ，シミュレーションにより生成した人流データと，街中に設置した微気象センサの実データを用いた評価実験の結果を示す．具体的には，まず時系列データの次元削減手法である SAX[12),19)] を拡張し，STAX と呼ばれる空間時系列の記号表現を導入する．次に，この表現に基づいて空間時系列を階層的に管理する木構造を構築し，この構造に適したアルゴリズムを用いて空間時系列の類似検索を行う．実験の結果，単純な総当たり方式で類似検索を行うと処理時間が 1 時間を超えるような場合であっても，I-Tree を用いればディスクアクセスの時間を含めて数秒以下で処理を完了することができ，索引の更新も高速に行えることが確認できた．また，従来手法よりも，問合せの長さやデータの内容にかかわらず，安定的に高速な類似検索を行えることが明らかになった．

2. 関 連 研 究

多数のセンサが埋め込まれた世界において有用なサービスを提供するためには，様々なデータ管理上の問題を解決しなければならない．センサに満ちた世界では，データの時空間的な傾向の理解を支援するサービスや，リアルタイムに特徴的なイベントや現象に反応できる機能が非常に重要である[5)]．このようなサービスや機能を円滑に提供するためには，センサデータに適した問合せ処理機構のデザインを行う必要があり，Madden らの TinyDB[14)] をはじめとして，様々な提案が行われてきた．特に，多数のセンサから次々と発生し続けるデータを扱うためには，永続的に蓄積されたデータを想定した処理機構を利用するだけでは不十分であるとの問題意識から，データストリームモデル[4),7)] に基づくシステムの研究が非常にさかんに行われている．COLR-Tree[2)] のように，問合せ処理を高速化するために空間索引機構に集約値のキャッシュを連携させる手法も提案されている．しかしながら，センサデータを対象とした従来型の問合せ処理機構は，様々なデータの時空間的なパターンを考慮した複雑な検索を高速に処理することが困難である．

時系列データのパターンを検索する手法はこれまでに多数提案されているが，特に時系列の類似度に基づく索引を用いる方式[1),8),19)] は，一般に検索結果を高速に求めることができる．時系列データに離散フーリエ変換を適用し，R-Tree 等の空間索引を用いて管理する方法[8)] については，雑音や波形の拡大縮小および平行移動を考慮した実際的な手法も提案されている[1)]．そのほかにも様々な手法が存在するが，特に最近研究が進んでいる簡便な記号列表現を用いる手法[12),15)] は，データストリームマイニング[9)] における重要な技法の 1 つにもなっている．しかしながら，これらの手法は単一の時系列を対象とするものであり，空間時系列を直接取り扱うことができない．

多くの場合，利用者は個々のセンサの時系列データよりも，複数のセンサによって観測される現象に興味を持っている[13)]．このような利用者の興味を意識したフレームワークとして，結合，選択，およびグループ化の処理を順に適用して時空間的なデータパターンに基づく現象の検知と追跡を可能にする PDT（Phenomena Detection and Tracking）[3)] がある．興味の対象が人流であれば，GPS 等を用いて個人の移動履歴データを収集し，これをまとめて処理する方法も考えられる．この場合，移動オブジェクトのデータ管理技術を利用することができる．たとえば，移動する多数のオブジェクトの状況を要約するために町田ら[25)] はヒストグラムに基づく手法を提案しており，また Kalnis ら[11)] は移動オブジェクトのクラスタを検出する手法について議論している．しかしながら，現状ではプライバシ等の問題のため，網羅的に人の移動履歴データを収集することは多くの場合困難である．固定設置型のセンサにより匿名の空間時系列データを収集し，これに基づいて人流の検出や予測を行う手法についても研究を行う必要がある．また，空間時系列の処理機構を用いれば，温度や湿度のような移動オブジェクトではないデータを扱うこともできる．

複数のセンサから発せられる時空間データを効率良く検索するための手法として，Papadias らの提案する時空間データウェアハウスの索引機構[17)] がある．この手法は，効率良くセンサのデータを空間的および時間的に集約することを可能にするものである．また，時空間データを効率良く連続的に監視するアルゴリズムである SINA[16)] は，時空間データに対する複数の問合せをまとめて効率良く処理する方式を用いている．しかしながら，これらは個々の属性値や集約値の検索を対象としており，系列データの類似検索を高速化する我々の提案手法に対して補完的な関係にある技術と位置付けることができる．SPIRIT[18)] と呼ばれる手法では，複数の時系列データをまとめてコンパクトに扱うために，主成分分析を利用している．また，近年提案された等高線地図のマッチング処理によって時空間的なパターンの検索を可能にする手法[21)] では，索引等を使わない逐次的な処理を行うため大量の空間時系列データを高速に処理することが難しい．もちろん，現象の検知だけでなく，一般的な時

間属性を持った空間データの管理[24] を効率良く行えることも重要である．おそらく I-Tree に最も近い手法は，Zhang らの提案する空間時系列の索引機構[22] である．Zhang らの手法では，空間自己相関に基づく領域分割によって構成される 4 分木を用いて，空間時系列の類似検索の高速化を行う．これに対して，我々の提案手法は記号配列表現に基づく索引を用いており，データの時空間的なパターンの違いに大きな影響を受けることなく安定して高速な処理を行うことができる．Zhang らの手法との比較については，5 章で詳しく議論する．

3. I-Tree

I-Tree が対象とするデータは，状態が時間とともに変化する空間データである．ただし，空間自体の位置や形状は変化しないものとする．空間的な広がりを持ったセンサデータから有意なパターンを高速に検索し，警報等のサービスをリアルタイムで提供するためには，以下の機能がセンサデータ管理機構に要求される．

(1)　空間時系列の類似検索を高速に処理できること．
(2)　空間時系列を高速に追加・削除できること．
(3)　索引を用いる場合は，短時間で索引を構築でき，主記憶容量に対して索引のサイズが大くなりすぎないこと．

これらの要求を満足させるために，時系列データマイニングで用いられる次元削減手法 SAX[12),19)] に基づく，空間時系列データの管理手法を提案する．

3.1 空間時系列のモデル

空間時系列とは，広義には時刻印と位置情報が付加されたデータの集合と考えてよいが，ここではいくつかの適当な前提を設けて議論を進めることにする．

まず，地理空間を格子状に分割して，各セルの時系列を集合的に扱うものとする．つまり，矩形の地理空間 R_A を検索対象領域とすると，R_A を x および y 方向に n，m 等分し，$n \times m$ 個のセルについてそれぞれ時系列データをまとめて取り扱う．ここで，x，y 軸に対して i，j 番目のセルの値を v_{ij}（$1 \leq i \leq n$，$1 \leq j \leq m$）と表記する．v_{ij} の値は，対応するセルに冗長にセンサが設置されている場合にはそれらのセンサの値の平均値を用いるものとし，必要なセンサが存在しないセルの場合は null とするか，適当な補間アルゴリズムを用いて値を充填する．

セル (i, j) の長さ z の時系列を $T_{ij} =< v_{ij1}, v_{ij2}, \ldots, v_{ijz} >$ とする．なお，$n \times m$ 個の時系列 T_{ij} は同期がとれているものとする．同期がとれていないデータを扱う場合は，補間等の前処理を行って時間を揃えておく．$n = m = 4$，$z = 16$ の場合の空間時系列データ **T** を**図 1** に示す．

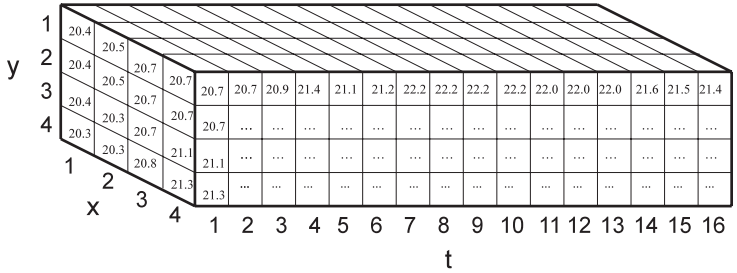


図 1　空間時系列データの例
Fig. 1　Sample spatial time-series data.

3.2 時空間断片集約近似

空間時系列は，一般に非常に次元数の多いデータである．したがって，空間時系列データを効率良く処理するためには，データの次元数を削減することが重要になる．そこで我々は，まず空間時系列データを PAA（Piecewise Aggregate Approximation：断片集約近似）[12)] と呼ばれる時系列データの次元削減手法に基づいて変換する．

時系列データを効率良く取り扱うための表現形式としては，PAA 以外にも，ウェーブレット変換や離散フーリエ変換等を用いた手法[1),8)] が多数提案されているが，PAA による表現であれば SAX（Symbolic Aggregate approXimation：記号集約近似）[12)] と呼ばれる離散的な記号列に容易に変換することができる．記号列として取り扱うことにより，既存の様々な文字列処理アルゴリズムやツールを容易に適用できる．したがって，様々なアプリケーションの要求に応じて有用な異種センサデータ検索サービスを提供するうえで，非常に好都合である．

空間時系列データを PAA 表現に変換するにあたって，まず部分空間時系列の切出しを行う．**図 2** に示すように，セルの時系列データ T_{ij} から，開始位置 t および終了位置 $t+l$ を 1 つずつ先に進めながら，長さ l の部分時系列 T_{ij}^t（$1 \leq t \leq z-l+1$）を順に切り出す．部分系列の長さ l については，興味のある現象の時間幅に基づいてあらかじめ決めておく．たとえば，5 分間の人流の変化に興味がある場合は，5 分単位で部分系列を切り出して索引を構築する．1 時間の人流の変化も検索したい場合は，別途 1 時間単位で部分系列を切り出して索引を構築することになる．なお，切り出した部分時系列といっしょに開始位置 t の情報を保存しておく．

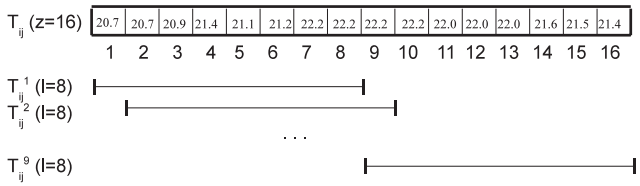


図 2　部分（空間）時系列の切出し
Fig. 2　Extracting partial spatial time-series data.

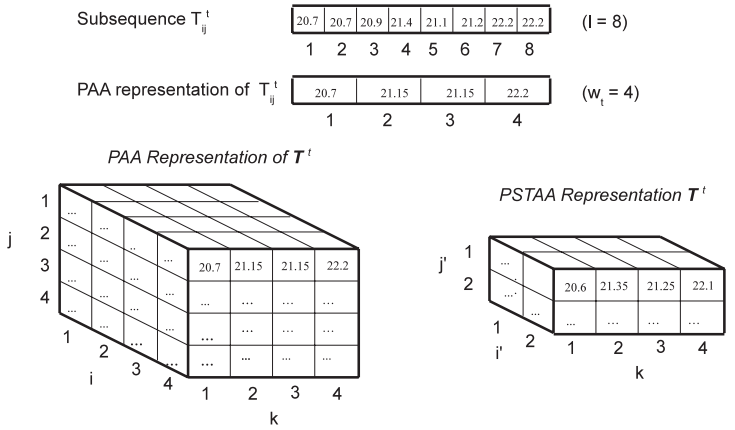


図 3　時空間断片集約近似表現（PSTAA）への変換
Fig. 3　Generating piecewise spatio-temporal aggregate approximation (PSTAA).

結果として，各セルの部分時系列データ集合 $T_{ij}^t =< v_{ij1}, v_{ij2}, \ldots, v_{ijl} >$（$1 \leq t \leq z-l+1$）が得られる．

次に，**図 3** に示すように，すべての部分時系列 T_{ij}^t の，PAA 表現 $\bar{T}_{ij}^t$ を求める．T_{ij}^t は l 次元であるが，その PAA 表現は w_t 次元である．一般に，w_t は l よりもかなり小さな数になる．

$$\bar{T}_{ij}^t =< \bar{v}_{ij1}^t, \bar{v}_{ij2}^t, \ldots, \bar{v}_{ijw_t}^t > \tag{1}$$

$$\bar{v}_{ijk}^t = \frac{w_t}{l} \sum_{s=\frac{l}{w_t}(k-1)+1}^{\frac{l}{w_t}k} v_{ijs}^t \tag{2}$$

x，y 軸方向についても同様に一定区間ごとの平均値を求めれば，$n \times m \times l$ 次元の部分空間時系列データを，$w_x \times w_y \times w_t$ 次元のデータに変換し，大幅に次元を縮減することができる．

$$\bar{v}_{i'j'k}^t = \frac{w_x w_y}{nm} \sum_{s_1=\frac{n}{w_x}(i'-1)+1}^{\frac{n}{w_x}i'} \sum_{s_2=\frac{m}{w_y}(j'-1)+1}^{\frac{m}{w_y}j'} v_{s_1 s_2 k}^t \tag{3}$$

部分空間時系列 $\mathbf{T}^t$ の時空間断片集約近似（Piecewise Spatio-Temporal Aggregate Approximation: $PSTAA$）表現 $\bar{\mathbf{T}}^t$ は，$\bar{v}_{i'j'k}^t$ を要素とする $w_x \times w_y \times w_t$ の配列である．なお，w_x，w_y，w_t のことを，PSTAA 表現の x，y，t 軸方向のワード長と呼ぶ．

3.3　時空間記号集約近似

SAX（Symbolic Aggregate approXimation：記号集約近似）[12] に基づき，$PSTAA$ による表現を離散的な記号列に変換する．まず，センサデータの値範囲を **a** 個の部分範囲に分割し，各部分範囲に適当な記号を割り当てる．なお，**a** を SAX 次数と呼ぶ．

部分空間時系列データ $\mathbf{T^t}$ に含まれるすべての値の平均と分散を μ，σ とする．ガウス曲線 $N(0,1)$ のつくる面積を **a** 等分する分割点の集合を $\{\beta_1, \beta_2, \ldots, \beta_{\mathbf{a}-1}\}$ とすれば，空間時系列データ $\mathbf{T}$ の次数 **a** による分割点は $b_i = \mu + \sigma\beta_i$（$1 \leq i \leq \mathbf{a}-1$）である．これらの点で分割された値範囲に，大きい順に 2 進数で番号を振り，対応する領域の記号表現として用いる．

$\mathbf{T}$ の部分空間時系列の $PSTAA$ 表現 $\bar{\mathbf{T}}^t$ の要素を $\bar{v}_{i'j'k}^t$ とし，$\bar{v}_{i'j'k}^t$ を含む領域の 2 進数記号を $bn_{i'j'k}^t$ とする．部分空間時系列 $\mathbf{T}^t$ の時空間記号集約近似（Symbolic Spatio-Temporal Aggregate approXimation: $STAX$）表現 $\bar{\mathbf{I}}^t$ は，$bn_{i'j'k}^t$ を要素とする $w_x \times w_y \times w_t$ の配列である．また，$STAX$ 表現 $\mathbf{I}^t$ の各要素を 10 進数に変換し SAX 次数 **a** を付したものを $\mathbf{N}^t$ とする（**図 4**）．各要素の SAX 次数は異なっていてもかまわない．

なお，2 つの STAX 表現の対応する記号の次数が異なる場合は，記号の 2 進数表現 $bn_{i'j'k}^t$ の長さが異なるが，この場合は先頭のビットから順番に比較を行い，短いほうの 2 進数表現を比較し終えた時点ですべてのビットが一致していれば，記号が一致したと見なす．

次に，Z 順序等に基づく空間充填曲線を用いて，記号配列 $\mathbf{N}^t$ を文字列に変換する．たとえば，$\mathbf{N}^t$ を Z 順序に基づいて $\{4^{16}, 5^{16}, 5^{16}, 2^8, 3^8, 3^8, 1^4, 6^{16}\}$ という系列に変換した後に，各要素を結合して 4.16_5.16_5.16_2.8_3.8_3.8_1.4_6.16 のように索引文字列 $iStr$ を作成する．**図 5** に示すように，索引文字列 $iStr$ とあわせて生の部分空間時系列データをデータベースに保存し，管理する．I-Tree を用いた検索では，索引の葉ノードに格納された $iStr$

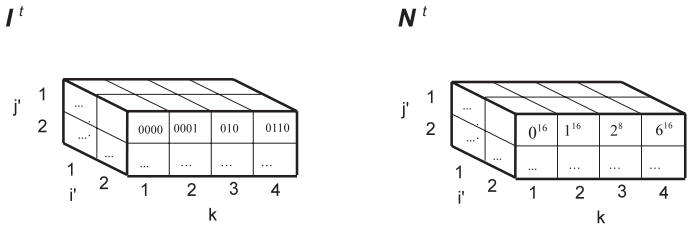


図 4　時空間記号集約近似表現（STAX）への変換
Fig. 4　Generating symbolic spatio-temporal aggregate approXimation (STAX).

iStr	Start time	Spatial Time Series Data
4.16_5.16_5.16_2.8_3.8_3.8_1.4_6.16	5	{20.1, 20.3, 20.1, 19.9, 20.0, 20.1, 20.2, 19.8, 19.9, 19.7, 20.0, 20.6, 20.8...., 21.2}
3.8_3.8_1.4_6.16_6.16_5.16_6.16_5.16	6	{20.5, 20.6, 20.5, 20.5, 20.5, 20.6, 20.7, 21.2, 21.3, 21.3, 21.3, 21.4, 21.3...., 22.2}
...	...	...

図 5　データベースへの格納
Fig. 5　Storing spatial time-series data in a database.

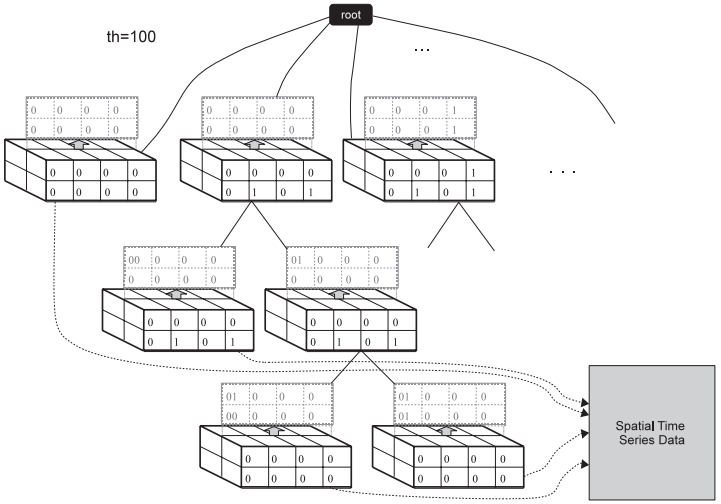


図 6　I-Tree の構成例
Fig. 6　Sample I-Tree.

を利用してデータベース中に格納された実際の空間時系列データにアクセスする．

3.4　I-Tree の作成

空間時系列データの検索を効率化するために，$STAX$ 表現に基づく索引である I-Tree を作成する．$STAX$ 表現 $\mathbf{I}^t$ の 1 つ以上の要素の次数を増加させると，$\mathbf{I}^t$ の表す空間時系列集合を要素に重なりのない 2 つの集合に分割することができる．データが増加するに従って適当な要素の次数を増加させ，**図 6** に示すような木構造によって分割管理する．

図 6 には，木構造の各ノードの STAX 表現が示されており，そのワード長は $w_x = 2$，$w_y = 2$，$w_t = 4$，閾値は $th = 100$ である．たとえば，ルートノード直下の左から 2 番目のノードに 100 個を超えるデータが挿入された時点で，ノードの分割が起こる．このノードの左上奥のセルの 2 進数の桁数を増やして子ノードを作成し，このセルの値の大小に基づいて親のノードの空間時系列を子ノードに振り分ける．以下同様の処理を繰り返して木を構築していく．

索引の作成にあたって，まず以下のパラメータの値を決めておく．

b：基本次数

w_x，w_y，w_t：ワード長

th：閾値

ここで，基本次数 b とは，ルートノード直下のノードの SAX 次数である．ルートノードの子ノードの最大数は $b^{w_x w_y w_t}$ であり，一般に b が大きいほど幅が広く高さの低い I-Tree が作成されやすくなる．

I-Tree には，以下の 3 種類のノードが存在する．

［葉ノード：］ 子ノードを持たない．葉ノードの iStr を用いてデータベースに保存された空間時系列データにアクセスすることができる．

［中間ノード：］ 子ノードを持つ．与えられた iStr に対応する子ノードを求めることができる．

［ルートノード：］ 中間ノードとほぼ同じである．ただし，ルートノードの子ノードの数は基本次数 b が大きいほど多くなる．

部分空間時系列データ $\mathbf{T}^t$ を I-Tree に挿入するアルゴリズムを以下に示す．なお，Node クラスの主要な変数とメソッドは，**表 1** に示すとおりである．

表 1　Node クラスの主要な変数とメソッド
Table 1　Relevant variables and methods in class Node.

変数名	説明	関数名	説明
Node parent	親ノードへのポインタ	getChild(searchIStr)	iStr が searchIStr に等しい子ノードを求める
Node[] children	子ノードへのポインタ	addChild(newNode)	newNode を子ノードとして追加する
boolean isLeaf	葉ノードとその他の区別	addData(T, stax, start)	開始時刻 start の系列 T をノードに登録する
boolean isRoot	ルートノードとその他の区別	deleteData(T, stax, start)	開始時刻 start の系列 T をノードから除外する
int level	深さ	getStax(T)	系列 T の STAX 表現を求める
int dataCount	管理中の系列数	getStartTime(T)	系列 T の開始時刻を求める
int childCount	子ノード数	getIStr(stax)	STAX を iStr に変換する
String iStr	iStr	getSearchIStr(level, stax)	STAX を木の深さに合った iStr に変換する（中間ノードの場合は，STAX の下位ビットを無視した iStr を作成する）
int[] startTime*	管理中の各系列の開始位置		
double dist*	検索処理中に距離を保存		
		insert(T, stax, start)	開始時刻 start の系列 T をノードに挿入する
		split()	ノードを分割する

* 必ずしもノードクラスに設ける必要がないオプショナル変数

Tree.insert($\mathbf{T}^t$, stax, startTime)

```
stax = getStax(T^t)
if root.hasChild() == false then
  newNode = new Node()
  root.addChild(newNode)
  newNode.iStr = getIStr(stax)
  newNode.isLeaf = true
  newNode.insert(T^t, stax, startTime)
else
  node = root
  while node.isLeaf == false do
    searchIStr = node.getSearchIStr(node.level, stax)
    node = node.getChild(searchIStr)
    if node is null then
      newNode = new Node()
      root.addChild(newNode)
      newNode.iStr = getIStr(stax)
      newNode.isLeaf = true
      node = newNode
    end if
  end while
  node.insert(T^t, stax, startTime)
end if
```

Node.insert($\mathbf{T}^t$, stax, startTime)

```
if this.dataCount is less than threshold then
  this.addData(T^t, stax, startTime)
else
  this.split()
  for all spatial time series T_c^t in this node do
    stax' = getStax(T_c^t)
    startTime' = getStartTime(T_c^t)
    searchIStr = getSearchIStr(this.level, stax')
    childNode = getChild(searchIStr)
    childNode.addData(T_c^t, stax', startTime')
    this.deleteData(T_c^t, stax', startTime')
  end for
  searchIStr = getSearchiIStr(this.level, stax)
  childNode = getChild(searchIStr)
  childNode.addData(T^t, stax, startTime)
end if
```

Tree クラスの insert メソッドを用いて，部分空間時系列 $\mathbf{T}^t$ を木に挿入する．$\mathbf{T}^t$ に対応する葉ノードが存在する場合には，Node クラスの insert メソッドを呼び出してこの部分空間時系列を葉ノードに追加する．対応する葉ノードがない場合には，ルートノードの直下に新たな葉ノードを作成することになる．なお，作成するノードの STAX 表現の各要素の SAX 次数は基本次数 b に等しい．

Node クラスの insert メソッドにおいては，葉ノードで管理する系列の数が閾値を超えた場合に split メソッドを用いてノードの分割を行う．この処理によって，現在のノードに 2 つの子ノードが生成される．子ノードの STAX 表現は，現在のノードの STAX 表現の SAX 次数を一部増加させ，空間時系列のより細かな弁別を可能にしたものである．次数を増加させる要素の選び方は自由であるが，ここでは簡単のため，ラウンドロビン方式で順番に 1

つずつ要素を選んで次数を増加させるものとする．次に，現在のノードに含まれているすべての空間時系列データ T_c^t について，STAX 表現（$stax'$），開始時刻（$startTime'$），新たな落ち着き先である子ノードを探すための iStr（$searchIStr$）を求め，見つかった子ノード（$childNode$）に T_c^t を登録し，現在のノードからは抹消する．

4. I-Tree を用いた検索処理

I-Tree を用いれば，問合せに対してユークリッド距離が最小となる系列を求める最近傍検索を高速に行うことができる．最近傍検索を高速に処理するために，問合せと時空間記号集約近似（STAX）が等しい空間時系列の集合をまず求める．

4.1 STAX 類似検索

問合せと時空間記号集約近似（STAX）が等しい空間時系列の集合を求めることを目的とした検索を，STAX 類似検索と呼ぶ．STAX 類似検索は，I-Tree を探索するだけで容易に処理することができる．つまり，新しい空間時系列データを I-Tree に挿入するときと同様に，空間時系列問合せの STAX 表現を求め，これに一致する葉ノードを探索すればよい．求められた葉ノードの iStr を用いてデータベースを検索し，得られた空間時系列をすべて出力する．このようにして出力される系列の数の最大値は閾値 th である．

稀に問合せの系列に一致する葉ノードが存在しない場合があり，途中で探索先が見つからなくなる．その場合は探索先が見つからなくなる直前のノードの配下にある葉ノードを求め，得られた空間時系列をすべて出力する．

4.2 最近傍検索

問合せとのユークリッド距離が最も近い系列を求める最近傍検索の処理においては，まず I-Tree を用いて STAX 類似検索を行う．STAX 類似検索の出力に最近傍検索の解が必ずしも含まれているとは限らないため，出力として得られた空間時系列データの中で問合せの系列に一番近いものを見つけ，その距離を現在までの処理過程で得られた次善の（Best So Far）最短距離を保存するための変数 BSF_dist に保存しておく．同様に，次善の最短距離の系列の開始時刻と，それに対応するノードの iStr を保持するための変数として BSF_startTime と BSF_iStr を用いる．

次に，問合せの系列と，各ノードに対応する系列集合について，距離の下限を近似的に求め，優先順位付きキューを用いて，距離の下限値が小さなノードから順番に処理を行う．変数 BSF_dist に保存された最小距離の値よりも距離の下限が大きなノードは無視することができるので，大幅に探索空間を小さくし，ディスクアクセスの回数を減らすことができる．

以下に示す最近傍検索のアルゴリズムでは，次の 4 つの関数を利用する．

(1) staxSearch($\mathbf{T}^t$)：STAX 類似検索を行う関数．
(2) getEuclideanMinDist($\mathbf{T}^t$, $istr$)：索引文字列 $istr$ に対応する空間時系列集合から，問合せの系列 $\mathbf{T}^t$ と最も近いものを求め，その距離を返す関数．
(3) minDistPstaa($\mathbf{T}^t$, node)：問合せの系列 $\mathbf{T}^t$ の $PSTAA$ 表現を用いて，$\mathbf{T}^t$ と任意のノード（に対応する空間時系列集合）の距離の下限を近似的に求める関数．
(4) getStartTimeOfNearestData($\mathbf{T}^t$, $istr$)：ノードの iStr の値が $istr$ であるとき，このノードで管理する系列の中で $\mathbf{T}^t$ と最も距離が近いものの開始時刻 t を求める関数．

関数 minDistPstaa では，問合せの系列 $\mathbf{T}^t$ の $PSTAA$ 表現 $\bar{\mathbf{T}}$ と任意の $STAX$ 表現 $\mathbf{I}^t$（の表現する可能な時系列集合）の距離の下限を求める関数 md を利用する．$md(\bar{\mathbf{T}}, \mathbf{I}^t)$ は次の式により計算することができる．

$$md(\bar{\mathbf{T}}, \mathbf{I}^t) = \sqrt{\frac{nml}{w_x w_y w_t}} \sqrt{\sum_{i',j',k} \begin{cases} (b^L_{i'j'k} - v_{i'j'k})^2 & if \;\; b^L_{i'j'k} > v_{i'j'k} \\ (b^U_{i'j'k} - v_{i'j'k})^2 & if \;\; b^U_{i'j'k} < v_{i'j'k} \\ 0 & otherwise \end{cases}} \tag{4}$$

$v_{i'j'k}$，$bn_{i'j'k}$ はそれぞれ $\bar{\mathbf{T}}$ と $\mathbf{I}^t$ の要素である．$v_{i'j'k}$ は数値であり，$bn_{i'j'k}$ は 2 進数の記号である．また，$v_{i'j'k}$ に対応する $bn_{i'j'k}$ の次数を $\mathbf{a}_{i'j'k}$ とする．$bn_{i'j'k}$ の上下のブレークポイントは $b^U_{i'j'k}, b^L_{i'j'k}$（$b^L_{i'j'k} < v_{i'j'k} < b^U_{i'j'k}$）で表す．ただし，$1 \le i' \le w_x, 1 \le j' \le w_y, 1 \le k \le w_t$ である．

I-Tree を用いた最近傍検索の処理手順は以下のとおりである．なお，node および minNode は Node クラスのインスタンスである．また，処理手順で用いている PriorityQueue は優先順位付きキューであり，決められた順序に従って要素を自動的に整列する機能を持っている．以下では，pq.add メソッドの呼び出しにより，I-Tree のノードをキューに追加しているが，このキューは問合せとノードの距離の下限に基づき整列された状態を保っているため，pq.extractMin メソッドを用いて距離の下限が最小であるノードをキューから容易に取り出すことができる．

Tree.exactSearch($\mathbf{T}^t$)

BSF_iStr = staxSearch($\mathbf{T}^t$)

```
BSF_dist = getEuclideanMinDist(T^t, BSF_iStr)
BSF_startTime = getStartTimeOfNearestData(T^t, BSF_iStr)

PriorityQueue pq = new PriorityQueue
pq.add(root)

while pq is not empty do
  minNode = pq.extractMin()
  if minNode.dist ⟩= BSF_dist then
    break
  end if
  if minNode.isLeaf == true then
    tmp = getEucledianMinDist(T^t, minNode.iStr)
    if BSF_dist ⟩ tmp then
      BSF_dist = tmp
      BSF_iStr = minNode.iStr
      BSF_startTime = getStartTimeOfNearestData(T^t, minNode.iStr)
    end if
  else if minNode.isLeaf == false then
    for all node in minNode.children do
      node.dist = md(T^t, node)
      pq.add(node)
    end for
  end if
end while
Return BSF_startTime
```

5. 性能評価実験

I-Tree のプロトタイプを開発し，シミュレーションによって人工的に生成した人流データと，市街地に設置した微気象センサにより取得した実データを用いて，類似検索および索引構築・更新に要する時間を測定し，従来手法と性能の比較を行った．

実験に使用した計算機は，CPU（Intel Xeon 1.86 GHz）を 4 基備えた Dell Precision WorkStation 690 であり，主記憶は 4 GByte，ハードディスクドライブは SEAGATE ST3250310AS（250 GB，SATA300，7,200 rpm），オペレーティングシステムは Windows XP SP3 である．なお，開発には Java 言語および PostgreSQL を用い，時系列データを管理するテーブル（図 5）においては，属性 iStr 上に B 木の索引を設けた．

表 2　RWP データセット
Table 2　RWP datasets.

	RWP1	RWP2	RWP3	RWP4	RWP5	RWP6	RWP7	RWP8	RWP9	RWP10
長さ	1 日	2 日	3 日	4 日	5 日	6 日	7 日	8 日	9 日	10 日
データ数（件）	2.2 M	4.4 M	6.6 M	8.8 M	11 M	13 M	15 M	18 M	20 M	22 M

5.1 データセット

性能評価に使用したのは，(1) ランダムウェイポイントモデル[6] に基づくシミュレーションによって生成した大量の人流データ（RWP），(2) ソーシャルフォースモデル[10] に基づくシミュレーションによって生成した道路ネットワークや歩行者群の細かな動きを考慮した人流データ（SFM），(3) そして群馬県館林市に微気象センサ[23] を設置して取得した実データ（MWS）の 3 種類のデータセットである．以下，それぞれのデータセットについて説明する．

5.1.1 ランダムウェイポイントモデルにより生成した人流データ（RWP）

1 km 四方の空間を 16 × 16 の格子状に分割し，約 63 m 四方の各領域に人数計測センサを設置した場合を想定し，シミュレーションにより 10 種類の空間時系列データを生成した．具体的には，まず障害物のない 1 km 四方の空間を 1,000 人の歩行者が自由に歩き続けるシナリオを想定し，ランダムウェイポイントモデル[6] を用いて，10 秒ごとの各歩行者の位置を記録した移動軌跡データを生成した．次に，この移動軌跡データを用いて，各時刻における 16 × 16 個の領域内の人数を合計し，空間時系列データを生成した．なお，人の平均的な歩行速度を考慮して，ランダムウェイポイントモデルにおける停止時間，最大速度，最小速度を，それぞれ 60 s，1.6 m/s，1.0 m/s とした．**表 2** に，各データセットの長さとデータ数を示す．

5.1.2 ソーシャルフォースモデルにより生成した人流データ

ランダムウェイポイントモデルを用いれば高速に大量の人流データを生成できるが，道路構造や歩行者同士のインタラクションを考慮していないため，このモデルにより計算される人の動きは，市街地等における実際の人の動きとはかなり異なるものとならざるをえない．

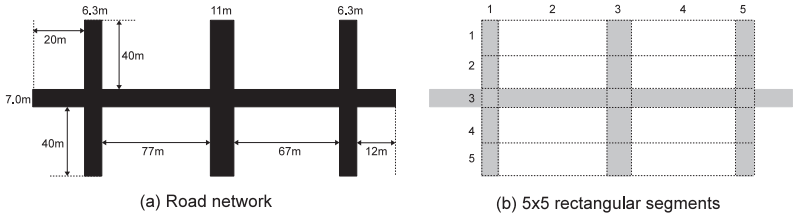


図 7　ソーシャルフォースモデルによる人流シミュレーションに利用した道路網
Fig. 7　Road network used in our Social Force Model-based simulation.

表 3　SFM データセット
Table 3　SFM datasets.

	SFM1	SFM2	SFM3	SFM4	SFM5
歩行者数	50	100	200	300	400

そこで我々は，実際の人の動きに近い人流データを生成するために，ソーシャルフォースモデル[10] に基づくシミュレーションを行った．このモデルは道路構造や歩行者同士のインタラクションを考慮したものであり，各歩行者の目標速度と行き先を与えれば，道路の境界や他の歩行者から受ける反発力や，特定の対象物から受ける引力を考慮して人流を計算することができ，道路が混雑するに従って人々がかたまって自然に流れを形成していく様子等も創発的なパターンとして出現させることができる．なお，歩行者の目標速度が平均 1.3 m/s，標準偏差 0.3 m/s のガウス分布に従うことを仮定し，人流を計算するために必要なその他のパラメータについても交通工学分野の知見に基づいて決定した[20]．

図 7 (a) にシミュレーションに使用した道路網モデルを示す．東京都内のある市街地の構造に基づくこの道路網モデルに 50 人，100 人，200 人，300 人，400 人の歩行者を注入し，1 時間分の移動軌跡データを 5 種類作成した．なお，各歩行者の位置は 1 秒ごとに記録した．次に，この道路網を図 7 (b) に示すように，道路に沿って 5×5 の領域に分割し，各時刻における各領域の人数を算出した．**表 3** に示すのが，このような手順によって生成した 5 つのデータセットである．なお，各データセットのデータ数は，領域数（5×5）と計測回数（1 秒ごとに 1 時間計測した場合 3,600 回）のみから算出可能であり，歩行者数にかかわらずいずれも 90,000（$= 5 \times 5 \times 3,600$）である．

5.1.3　微気象センサにより取得した温度データ（MWS）

我々は群馬県館林市に微気象センサネットワークを展開し，2009 年夏から継続的に気温と湿度の計測を行っている[23]．館林駅東側の 600 m 四方の市街地領域に数 10 m 間隔で 44 個の気象センサを設置し，1 分間隔でセンシングを行っている．センサノード間の通信には IEEE 802.15.4 を使用し，マルチホップ方式で転送されたデータはシンクノードを経由してサーバに集められる．このセンサネットワークによって収集した 1 週間分（2009 年 8 月 11 日 0 時 00 分から同年 8 月 17 日 23 時 59 分までの）の温度データを用いて，空間時系列データを生成した．この市街地領域を格子状に 2×2 の領域に分割し，各時刻における各領域の平均温度を算出した．また，欠損したデータについては前後の時刻のデータを基に線形補間を行った．なお，このデータセット（MWS）のデータ数は 40,278 個である．

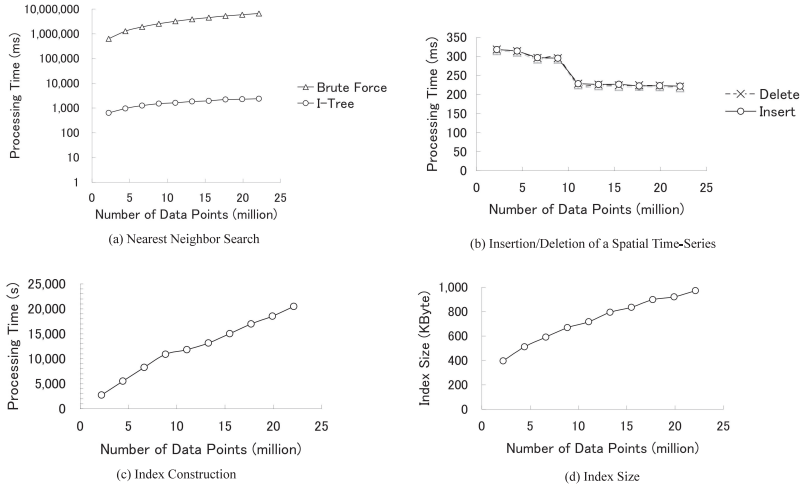


図 8　データ数の増加にともなう I-Tree の性能の変化
Fig. 8　Impact of data size on processing time and index size of I-Tree.

5.2　大量データの処理における I-Tree の性能

既存手法との比較を行う前に，まず大量データの処理における I-Tree の基本的な特徴を理解するために，データセット RWP1, . . . , RWP10 を用いて I-Tree を構築し，長さ 128 の空間時系列問合せを対象として最近傍検索に要する時間を計測した．なお，基本次数を 2，ワード長 $w_x = 2$，$w_y = 2$，$w_t = 4$，閾値 100 の I-Tree を用いるとともに，各データセットから異なる時刻を起点とする部分空間時系列を 10 個抽出して問合せの系列として利用し，処理時間の平均値を計算した．

図 8 (a) に最近傍検索の処理時間を示す．ただし，処理時間は対数軸を用いて示してい

る．総当たり（Brute Force）で最近傍検索を行う場合，データ数が 2,200 万程度の場合（RWP10），処理時間の平均は 2 時間弱であったが，I-Tree を用いた場合の処理時間はディスクアクセスの時間を含めて 2.4 秒以下であった．なお，I-Tree のノードを探索して対応する iStr を得るのに要する時間は，1〜2 ms 以下であり，得られた iStr をキーとして B 木索引を用いてデータベースから目的の空間時系列を取得する時間は数百 ms 秒前後であった．

空間時系列の挿入および削除が行われた際に I-Tree を管理するための処理時間は，データ数にかかわらず 200〜300 ms 程度であるという結果が得られた（図 8 (b)）．なお，データ数が比較的少ない場合には，ノードの分割や結合が生じやすいため多少処理時間が長くなっていると考えられる．

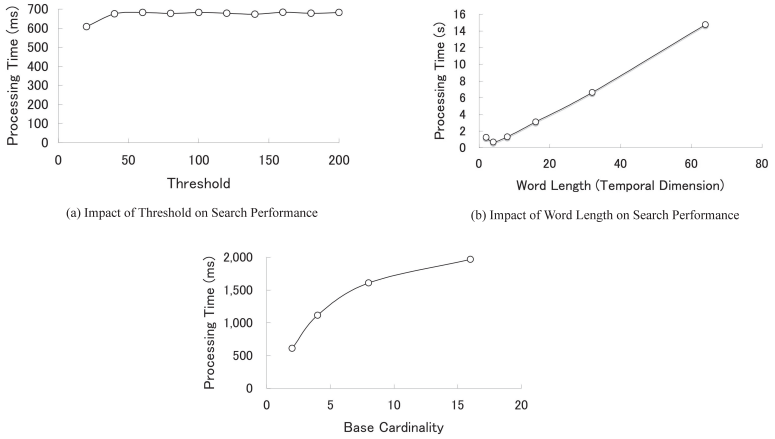
I-Tree を構築するのに要する時間と，構築された索引のサイズを図 8 (c) および図 8 (d) に示す．索引構築時間および索引サイズはデータ数にほぼ比例するかたちで増加している．索引サイズについてはデータ数が 2,200 万程度の場合（RWP10）で 1 MByte 程度であった．この程度のサイズであれば主記憶中に保持することが可能であるが，I-Tree を主記憶に保持しきれなくなるほどの莫大なデータを扱う場合には，索引探索時のディスクアクセスによって処理速度が低下することが予想される．

図 9 は，I-Tree のパラメータ（閾値，ワード長，基本次数）の増加にともなう最近傍検索時間の変化を示したものである．使用したデータセットは RWP1 である．図 9 (a) に示すとおり，今回使用したデータセットの場合，閾値を 20〜200 の範囲で変化させても検索時間に大きな変化はないという結果を得た．これに対して，図 9 (b) および (c) に示すとおり，ワード長と基本次数は検索時間に明確な影響を与えている．ワード長については，$w_x = 2$，$w_y = 2$ とし w_t を 2 から 64 まで変化させたが，$w_t = 4$ の場合，最も高速な処理が可能であり，w_t をそれより小さくすると（$w_t = 2$）処理時間がやや増大している．

5.3　従来手法との比較

I-Tree の性能をより詳細に分析するために従来手法との比較評価を行った．時空間データの統合的な検索を高速化する従来手法の中で，I-Tree と同様に索引を用いて空間時系列の類似検索を高速化する Zhang らの手法[22] を比較対象として，最近傍検索の処理時間，索引構築の処理時間，索引のサイズを分析した．なお，検索処理時間の計測においては，検索対象である空間時系列から適当な部分系列を 10 個取り出して，これらに最も距離の近い系列を求め，処理時間の平均を算出した．

Zhang らの手法は，地理的に近い空間で観測されるデータは似ていることが多いという仮定に基づいて，観測量の相関性が強い隣接領域をまとめて管理する 4 分木による索引構造を



(a) Impact of Threshold on Search Performance

(b) Impact of Word Length on Search Performance

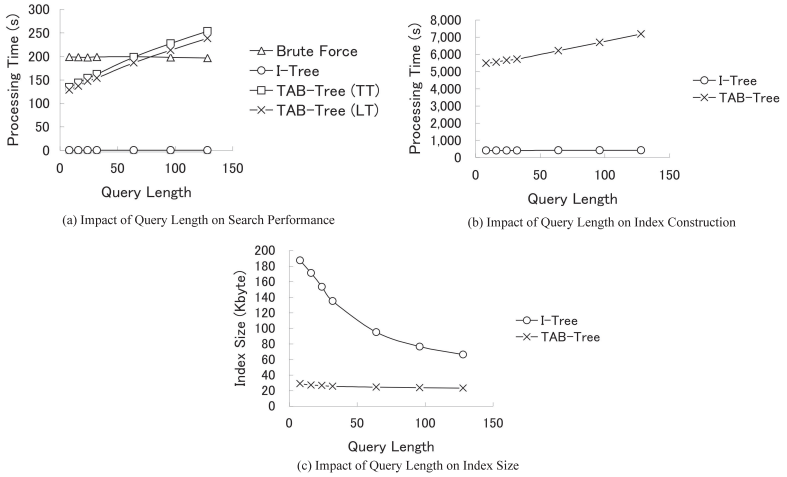
(c) Impact of Base Cardinality on Search Performance

図 9　閾値，ワード長，基本次数の増加にともなう I-Tree の最近傍検索時間の変化

Fig. 9　Impact of threshold values, word lengths, and base cardinalities on search performance of I-Tree.

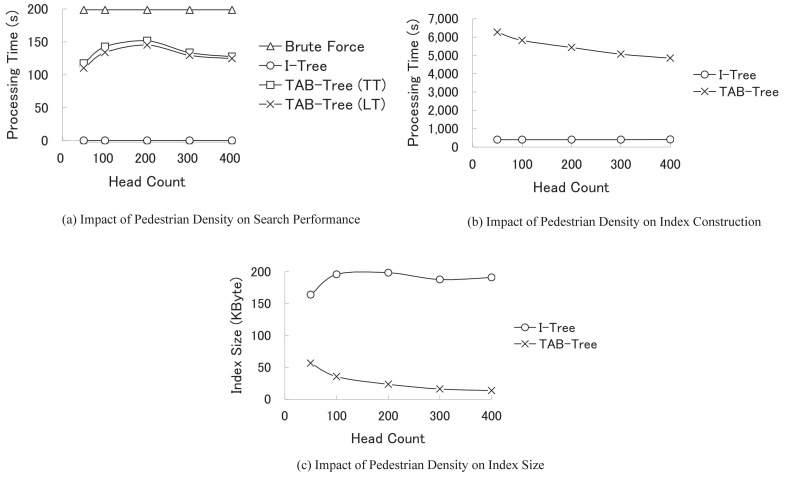
導入し，類似検索の高速化を行うものである．基本的にはこの 4 分木は時系列の空間的な相関性を扱うものである．しかしながら，同じ地理領域の過去の人流データを検索して混雑の予測を行うような場合には，同じ空間の部分空間時系列データを開始時刻を変えながら比較する必要があるため，空間的な相関性に基づく索引を用いても高速化が期待できない．そこで我々は，Zhang らの手法に基づいて時間方向のデータの相関性を扱うために，前述の 4 分木とほぼ同じ方法で，相関性に基づいて時間方向にデータを分割する 2 分木による索引機構を実装した．以下ではこの索引機構を便宜上 TAB-Tree（Temporal Autocorrelation-Based Search Tree）と呼ぶことにする．Zhang らはルートノードから葉ノードへと木を探索して処理を進める方式と，葉ノード間のリンクを用いて直接葉ノードから葉ノードへと飛びながら処理を進める方式を提案している．これらを区別するために，前者の検索処理方式を TT（Tree-based Traversal），後者を LT（Leaf-based Traversal）と表記することにする．

このように，I-Tree との比較をできるだけ公平に行うために Zhang らの手法を拡張した．しかしながら，もともと両索引機構が対象とする検索の種類や類似度の定義は異なるため，TAB-Tree を用いた最近傍検索の結果と I-Tree を用いた最近傍検索の結果が多少異なる場合がある．具体的には，I-Tree がユークリッド距離を用いて類似度を判定するのに対して，



(a) Impact of Query Length on Search Performance
(b) Impact of Query Length on Index Construction
(c) Impact of Query Length on Index Size

図 10　問合せ長の増加にともなう性能の変化
Fig. 10　Impact of query length on processing time and index size.



(a) Impact of Pedestrian Density on Search Performance
(b) Impact of Pedestrian Density on Index Construction
(c) Impact of Pedestrian Density on Index Size

図 11　歩行者数の増加にともなう性能の変化
Fig. 11　Impact of pedestrian density on processing time and index size.

Zhang らの手法ではコサイン距離を用いて類似度を判定している．また，Zhang らの手法が主に対象とするのは最近傍検索でなく範囲検索であるため，問合せからの距離が適当なコサイン距離（< 0.1）以下である系列を範囲検索により求めて解の候補を絞り込み，得られた候補の中から問合せと最も近い系列を求める手続きを実装した．今回の実験では問合せと最近傍検索の解の最大距離が既知であったため，この手続きによってつねに最近傍検索の解を求めることができた．さらに付加的な処理を行えば，Zhang らの手法を用いた最近傍検索の結果を I-Tree による検索結果に近づけることができるかもしれないが，付加処理により必要な記憶領域や処理時間を考慮する必要があり，むしろ I-Tree の相対的な優位性を高める結果になると考えられる．

ソーシャルフォースモデルにより生成したシミュレーションデータと微気象センサを用いて取得した温度データを用いて，I-Tree と TAB-Tree の性能を比較した．

表 3 に示す 5 つのデータセットを用いた比較評価の結果を**図 10** に示す．使用した I-Tree のワード長は $w_x = 2$，$w_y = 2$，$w_t = 4$，閾値は 100，基本次数は 2 である．また，TAB-Tree の構築においては，ノード内のデータの相関性を表す指標である円錐の角度のコサイン値が 0.9 以上の場合にノードを分割した．索引を用いた処理の対象となる問合せの系列長を 8 から 128 まで変化させた場合の処理時間と索引サイズを示しているが，これらは 5 つのデータセットの平均値である．最近傍検索の処理時間については，図 10 (a) に示すとおりである．TAB-Tree の場合，問合せの系列長が長くなるにつれて処理時間が増加しているが，I-Tree の場合は問合せの系列長にかかわらずつねに短時間で検索処理を完了することができた．索引構築の処理時間については，図 10 (b) に示すとおりである．TAB-Tree を構築するためには，空間時系列の相関性を示す指標を計算する必要があり，このための計算処理に多くの時間が費やされている．図 10 (c) に示すように，索引サイズについては全般的に TAB-Tree のほうが小さくなっている．また，I-Tree の索引サイズは，問合せの系列長が増加するに従って減少している．

登録された系列の数が閾値 th より非常に小さなノードが多数存在する場合，I-Tree の総ノード数が増加し，索引サイズが増大する．本実験に用いたデータセットの場合，問合せの系列が長いほどデータが疎なノードが減少し，総ノード数も減少したため，結果として索引サイズも減少することになったと考えられる．なお，ここでいう索引サイズとは，表 1 におけるオプショナル変数を除く変数が必要とする記憶領域を全ノードについて合計したものである．なお，オプショナル変数とした dist と startTime は，必ずしもノード内に保持する必要はなく，プログラム中の別の場所やデータベースで管理してもかまわない．

図 11 は，データセット SFM1, SFM2, . . . , SFM5 それぞれについて，I-Tree と TAB-Tree

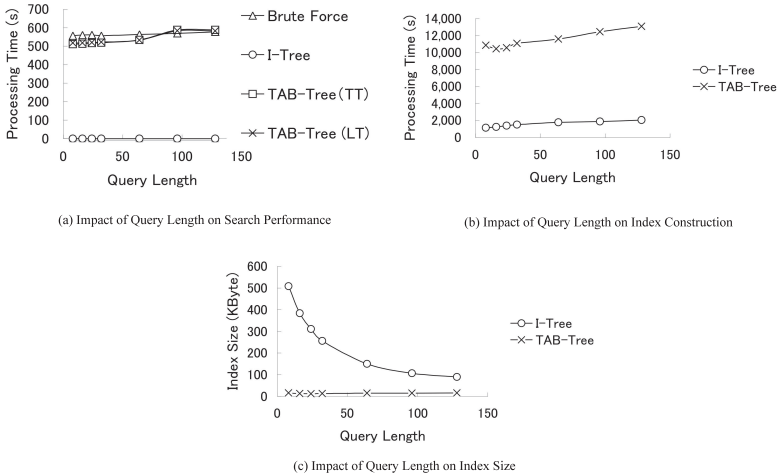


図 12　実データを用いた性能の比較
Fig. 12　Performace evaluation using real sensing data.

の性能を比較したものである．先と同様に，I-Tree のワード長は $w_x = 2$，$w_y = 2$，$w_t = 4$，閾値は 100，基本次数は 2 とし，TAB-Tree の構築における円錐角のコサインの値の閾値は 0.9 とした．また，問合せの系列長は 8 とした．図 11 (a) に示すように，TAB-Tree の検索処理時間はソーシャルフォースモデルを用いたシミュレーションにおける歩行者数に影響を受ける．歩行者数が増加するにつれて混雑が生じ，データの空間的および時間的なパターンが変化する．この変化が TAB-Tree の検索性能に影響を及ぼしているのではないかと考えられる．これに対して，I-Tree を用いた場合は，歩行者数が増加しても検索処理をつねに短時間で行うことができる．また，図 11 (b) および図 11 (c) には，先と同様に，索引構築時間は I-Tree のほうが短かく，索引のサイズは TAB-Tree のほうが小さいという結果が示されている．5.1.2 項で示したように，歩行者数にかかわらずデータ数は一定であるが，人数が異なると異なる混雑パターンが創発的に出現する．人数が非常に少ない場合や非常に多い場合は，時空間的なパターンが比較的一様になる傾向が見られるため，データが疎なノードが減少し，総ノード数および索引サイズがやや小さくなるのではないかと考えられる．

最後に，市街地に微気象センサを設置して取得した実データを用いて性能の比較を行った結果を**図 12** に示す．なお，I-Tree のパラメータについては先と同様で，ワード長は $w_x = 2$，$w_y = 2$，$w_t = 4$，閾値は 100，基本次数は 2 である．本データは人流データと異なるパターンを示しており，TAB-Tree の性能を十分に引き出すためには，TAB-Tree の構築における円錐角のコサインの値の閾値を 0.9 よりも高い値にすべきであることが判明した．そこで閾値として 0.9995 を採用し，また問合せの系列長を 128 とした．人流データを対象とした場合の検索性能（図 10 (a)）と実測した温度データを対象とした場合の検索性能（図 12 (a)）を比較すると，TAB-Tree の処理時間の増加は，後者の方がやや緩やかであることが分かる．I-Tree では，実測した温度データの場合もつねに短時間で検索処理を行うことができた．図 12 (b) および図 12 (c) に示すように，索引構築時間と索引サイズについては，実測した温度データの場合も，人流データの場合と同様の傾向を示している．

6. むすび

空間時系列データの高速な類似検索を可能にする索引機構 I-Tree を提案し，ランダムウェイポイントモデルおよびソーシャルフォースモデルに基づく人工的な人流データと，市街地に設置した微気象センサを用いて実測した温度データを用いて性能の評価を行った．

I-Tree は，従来手法に比べて，様々な系列長の問合せや変化パターンの異なるデータに対して，安定的に高速な類似検索処理が可能であるという特徴を持ち，データ量が増加しても索引を更新するコストが目立って増加することはなかった．索引のサイズは従来手法より大きくなるが，データ数が 2,200 万の場合で 1 MByte 程度であり，さらにデータ数が増えても索引を主記憶中に保持して処理を行うことができると考えられる．蓄積された空間時系列の量が増えてくると，処理速度の遅い総当たり方式では混雑を予測してタイムリに警報を発信するといったシナリオを実現することが難しくなるが，このような場合でも I-Tree を用いればシナリオを実現するに十分な高速性を達成できると考えている．また，様々なデータや問合せを安定的に高速処理できる索引機構を実現できれば，異なるセンサから取得したデータの処理を同じ索引機構で高速化することが可能になり，異種センサデータの統合処理環境を実現するうえでも有用であると考えている．

参考文献

1) Agrawal, R., Lin, K.-L., Sawhney, H.S. and Shim, K.: Fast Similarity Search in the Presence of Noise, Scaling and Translation in Time-Series Databases, *Proc. Int'l. Conf. Very Large Data Bases*, pp.490–501 (1995).
2) Ahmad, Y. and Nath, S.: COLR-Tree: Communication Efficient Spatio-Temporal Index for a Sensor Data Web Portal, *Proc. Int'l. Conf. Data Engineering*, pp.784–793 (2008).

3) Ali, M.H., Mokbel, M.F., Aref, W.G. and Kamel, I.: Detection and Tracking of Discrete Phenomena in Sensor-Network Databases, *Proc. Int'l. Conf. Scientific and Statistical Database Management*, pp.163–172 (2005).
4) Babcock, B., Babu, S., Datar, M., Motwani, R. and Widom, J.: Models and issues in data stream systems, *Proc. ACM SIGMOD-SIGACT-SIGART Symposium Principles of Database Systems* (*PODS*), pp.1–16 (2002).
5) Balazinska, M., Deshpande, A., Franklin, M.J., Gibbons, P.B., Gray, J., Nath, S., Hansen, M., Liebhold, M., Szalay, A. and Tao, V.: Data Management in the Worldwide Sensor Web, *IEEE Pervasive Computing*, Vol.6, No.2, pp.10–20 (2007).
6) Camp, T., Boleng, J. and Davies, V.: A Survey of Mobility Models for Ad Hoc Network Research, *Wireless Communications and Mobile Computing*, Vol.2, No.5, pp.483–502 (2002).
7) Carney, D., Çetintemel, U., Cherniack, M., Convey, C., Lee, S., Seidman, G., Stonebraker, M., Tatbul, N. and Zdonik, S.: Monitoring streams: A new class of data management applications, *Proc. Int'l. Conf. Very Large Data Bases*, pp.215–226 (2002).
8) Faloutsos, C., Ranganathany, M. and Manolopoulos, Y.: Fast Subsequence Matching in Time Series Databases, *Proc. ACM SIGMOD Int'l. Conf. Management of Data*, pp.419–429 (1994).
9) Gaber, M.M., Zaslavsky, A. and Krishnaswamy, S.: Mining Data Streams: A Review, *SIGMOD Record*, Vol.34, No.2, pp.18–26 (2005).
10) Helbing, D., Buzna, L., Johansson, A. and Werner, T.: Self-Organized Pedestrian Crowd Dynamics: Experiments, Simulations, and Design Solutions, *Transportantion Science*, Vol.39, No.1, pp.1–24 (2005).
11) Kalnis, P., Mamoulis, N. and Bakiras, S.: On Discovering Moving Clusters in Spatio-temporal Data, *Advances in Spatial and Temporal Databases: Proc. Int'l. Symposium on Spatial and Temporal Databases* (*SSTD 2005*), Lecture Notes in Computer Science, Vol.3633, pp.364–381, Springer, London (2005).
12) Lin, J., Keogh, E., Wei, L. and Lonardi, S.: Experiencing SAX: A novel symbolic representation of time series, *Data Mining and Knowledge Discovery*, Vol.15, No.2, pp.107–144 (2007).
13) Luo, L., Kansal, A., Nath, S. and Zhao, F.: Sharing and Exploring Sensor Streams over Geocentric Interfaces, *Proc. ACM SIGSPATIAL Int'l. Conf. Advances in Geographic Information Systems* (2008).
14) Madden, S.R., Franklin, M.J., Hellerstein, J.M. and Hong, W.: TinyDB: An Acquisitional Query Processing System for Sensor Networks, *ACM Trans. Database Systems*, Vol.30, No.1, pp.122–173 (2005).
15) Megalooikonomou, V., Wang, Q., Li, G. and Faloutsos, C.: A Multiresolution Symbolic Representation of Time Series, *Proc. Int'l. Conf. Data Engineering*, pp.668–679 (2005).
16) Mokbel, M.F., Xiong, X. and Aref, W.G.: SINA: Scalable Incremental Processing of Continuous Queries in Spatio-temporal Databases, *Proc. 2004 ACM SIGMOD Int'l. Conf. Management of Data*, pp.623–634 (2004).
17) Papadias, D., Tao, Y., Kalnis, P. and Zhang, J.: Indexing Spatio-Temporal Data Warehouses, *Proc. 18th Int'l. Conf. Data Engineering*, pp.166–175 (2002).
18) Papadimitriou, S., Sun, J. and Faloutsos, C.: Streaming Pattern Discovery in Multiple Time-Series, *Proc. Int'l. Conf. Very Large Data Bases*, pp.697–708 (2005).
19) Shieh, J. and Keogh, E.: iSAX: Indexing and Mining Terabyte Sized Time Series, *Data Mining and Knowledge Discovery*, Vol.19, No.1, pp.24–57 (2009).
20) Thepvilojanapong, N., Konomi, S. and Tobe, Y.: A Study of Cooperative Human Probes in Urban Sensing Environments, *IEICE Trans. Communications, Special Section on Fundamental Issues on Deployment of Ubiquitous Sensor Networks*, Vol.E93-B, No.11, pp.2868–2878 (2010).
21) Xue, W., Luo, Q., Chen, L. and Liu, Y.: Contour Map Matching for Event Detection in Sensor Networks, *Proc. ACM SIGMOD Int'l. Conf. Management of Data*, pp.145–156 (2006).
22) Zhang, P., Huang, Y., Shekhar, S. and Kumar, V.: Exploiting Spatial Autocorrelation to Efficiently Process Correlation-Based Similarity Queries, *Advances in Spatial and Temporal Databases: Proc. Int'l. Symposium Spatial and Temporal Databases* (*SSTD 2003*), Lecture Notes in Computer Science, Vol.2750, pp.449–468 (2003).
23) 戸辺義人，蔵田英之：細粒度気象センサネットワーク構築の実際—群馬県館林市の例，情報処理，Vol.51, No.6, pp.692–699 (2010).
24) 中村泰明，出来原裕順：時間属性をもった空間データの管理構造—PMD 木，情報処理学会論文誌：データベース，Vol.40, No.SIG 5 (TOD 2), pp.54–68 (1999).
25) 町田陽二，石川佳治，北川博之：マルコフ連鎖モデルに基づく動的な移動ヒストグラム構築手法，日本データベース学会 Letters，Vol.5, No.1, pp.89–92 (2006).
26) 木實新一，石塚宏紀，岩井将行，宮崎　純，戸辺義人：I-Tree：異種センサデータの統合利用を支援する複合型索引機構，マルチメディア，分散，協調とモバイル（DICOMO2010）シンポジウム予稿集，pp.92–99 (2010).
27) 木實新一，石塚宏紀，岩井将行，宮崎　純，戸辺義人：I-Tree：異種センサデータを用いた空間時系列検索の支援，信学技報（DE），Vol.110, No.107, pp.33–38 (2010).

(平成 22 年 9 月 20 日受付)
(平成 23 年 1 月　4 日採録)

(担当編集委員　喜田 拓也)

木實　新一（正会員）

平成 3 年九州大学大学院工学研究科情報工学専攻修士課程修了．同年九州大学大学院工学研究科助手．平成 6 年京都大学大学院工学研究科助手．平成 8 年工学博士．以後，ドイツ国立情報処理研究所，コロラド大学計算機科学科，東京大学空間情報科学研究センター，東京電機大学未来科学部にて，主として HCI，Ubicomp，CSCW，DB 分野の研究に従事．電子情報通信学会，日本データベース学会，ACM，IEEE CS 各会員．

石塚　宏紀（学生会員）

平成 17 年東京電機大学工学部情報メディア学科卒業．平成 19 年同大学大学院修士課程修了．平成 21 年東京大学大学院情報理工学系研究科博士課程入学．ユビキタスコンピューティング，センサネットワークの研究に従事．

岩井　将行（正会員）

平成 14 年慶應義塾大学大学院政策・メディア研究科修士課程修了．現在，東京大学生産技術研究所助教．分散ミドルウェア，分散イベント通信，情報家電協調制御等の研究に従事．

宮崎　　純（正会員）

奈良先端科学技術大学院大学情報科学研究科准教授．博士（情報科学）．平成 4 年東京工業大学工学部情報工学科卒業．平成 9 年北陸先端科学技術大学院大学情報科学研究科博士後期課程修了．同大学助手を経て，平成 15 年より現職．平成 12〜13 年テキサス大学アーリントン校客員研究員．平成 15〜19 年科学技術振興機構さきがけ研究員．高性能・高機能データベースならびに情報検索の研究に従事．電子情報通信学会，日本データベース学会，ACM，IEEE CS 各会員．

瀬崎　　薫（正会員）

昭和 59 年東京大学工学部電気工学科卒業．平成元年同大学大学院博士課程修了．同年東京大学生産技術研究所講師．現在，東京大学空間情報科学研究センター准教授．平成 12〜15 年国立情報学研究所客員助教授．平成 13 年より電気通信事業紛争処理委員会特別会員．平成 8〜9 年 UCSD 客員研究員．通信ネットワーク，ロケーション&コンテクスト・アウェアネットワークサービス，高速スイッチシステム，GIS の研究に従事．工学博士．平成 2 年篠原記念学術奨励賞受賞．IEEE 会員．

戸辺　義人（正会員）

東京電機大学未来科学部情報メディア学科教授．独立行政法人科学技術振興機構 CREST OSOITE プロジェクトにて，アーバンセンシングの研究を進めている．