M-037

大規模なプローブデータを活用した渋滞推定技術におけるマップマッチング手法
The Map Matching Method in Estimation of Traffic Condition using Large-scale probe data

竹内 智晴† 平田 飛仙† 菅野 幹人† 飯塚 剛†
Tomoharu Takeuchi† Takahisa Hirata† Mikihito Kanno† Tsuyoshi Iizuka†

1. はじめに

近年の情報通信技術の発達に伴い、渋滞情報を収集して道路の渋滞状況を推定、予測するサービスが普及し始めている。渋滞情報は、道路上に設置された感知器よって車両の移動速度をリアルタイムに観測する VICS (Vehicle Information and Communication System)や、GPS (Global Positioning System)によって観測された車両の位置座標、瞬間速度、進行方向等により構成される車両データ(プローブデータ)を利用することにより、知ることができる。

VICS を用いた渋滞予測技術としては、過去 VICS の渋滞情報を蓄積し、将来の渋滞変化パターンを予測する技術が提案されている[1,2]。ただし、車両を観測する路上感知器の設置に膨大な設備コストが必要なため、VICS が提供する渋滞情報は高速道路や主要幹線道路などの一部道路にのみに限られている。そのため、網羅的な渋滞予測ができないことが問題である。一方、プローブデータを活用した渋滞予測としては、過去の渋滞パターンから現在プローブデータ情報を含めた渋滞状況を元に予測を行う手法[3]、プローブデータ履歴の主成分分析によって得られる特徴空間を用いて予測値を求める手法[4]などが提案されている。プローブデータを用いた渋滞情報は、車載器による車両観測のため設備コストはほとんど必要とせず、理想的には、あらゆる道路の交通情報を入手可能である。そのため、プローブデータ活用により、主要幹線道路以外の脇道も含めた網羅的な渋滞予測が期待できる。

プローブデータを渋滞予測に活用するには、車両ごとの位置や速度の情報であるプローブデータから、道路を交差点などの区分で分割したリンクと呼ばれる単位で、渋滞情報を推定する必要がある。そのためには、各プローブデータがどのリンクの走行データであるかを判定して、プローブデータとリンクとを対応付けるマップマッチングを行う。近年のプローブデータ活用技術の普及に伴って、プローブデータは年々大規模化している。これに対応しリアルタイムな渋滞情報を提供するためには、数十万台規模のプローブデータを、秒単位で処理する高速なプローブデータ処理技術が求められる。そこで、本稿において、プローブデータの大規模化に対応するためのプローブデータ処理技術として、高効率な新規のマップマッチング手法を提案する。

2. マップマッチングの高速化

2.1 既存手法

マップマッチングの最も単純な方法としては、プローブデータの位置座標と各リンクとの距離を計算して、距離最小となるリンクと対応付ける方法がある[5]。しかし、全国すべてのリンクに対して距離計算を行うと、膨大な計算が必要なため、リアルタイムに処理を行うことは困難である。

そこで、マップマッチングの高速化手法として、事前に地図領域を格子分割し、分割格子内のリンクとの距離計算のみに絞り込むことで、計算回数を削減して高速化する手法が提案されている[6]。この手法では、マップマッチングの処理速度は、格子サイズを小さくするほど、計算対象となるリンク本数を削減できるため、処理を高速にすることが可能となる。しかし、プローブデータの位置座標は GPS の測定誤差を含むため、格子サイズを小さくしすぎるとマップマッチングを誤る可能性が高まる。そのため、この手法では GPS 誤差や道路幅を考慮した格子サイズとし、さらにプローブデータの存在する格子に隣接する 8 つの格子領域も考慮してマップマッチングを行う。これにより、距離計算が必要なリンク数を高々数リンクまで絞り込むことができるため、処理速度を向上しつつ、かつ精度を落とさずにマップマッチングを行うことを可能とした。

2.2 課題

実際に存在するリンク(道路)は、そのほとんどが単純な直線ではなく、カーブを伴う曲線リンクである。そのため、単純に点と直線の距離計算ではリンクからの距離を求めることができない。通常、地図データ中でリンクを表現する場合には、複数の座標点(ノード)を連結させて表す。カーブがあるリンクとの距離計算を行うには、隣接ノードを結ぶ直線それぞれとの距離計算を行うなどの工夫が必要であり、その場合の計算時間はリンクを構成するノード数に依存する。一般道では平均数十ノード程度で、多い場合は百ノード以上になる。特にカーブの多いリンクや長いリンクなどはノード点数が多くなりやすく、そのようなリンクは距離計算に要する時間が増大する。よって、リンク形状(ノード点数)に計算時間が依存しないマップマッチング手法の確立が必要である。

3. 提案手法

3.1 フィルタ計算を用いたマップマッチング

そこで我々は、より効率的にマップマッチングを行うため、プローブデータとリンクとの距離やリンクの進行方向等を重みとして反映したフィルタを記述し、フィルタ演算によりマップマッチングを行う手法を提案する。

提案手法では、事前に距離や進行方向を計算しておき、リンク別にフィルタとして記述しておくことで、マップマッチング計算における距離計算を省略して処理の効率

†三菱電機　情報技術総合研究所

化を図る。距離計算をフィルタ計算に置換することにより、リンク形状の複雑化に伴う計算の増加を取り除き、マップマッチング処理を高速に行うことを可能にする。このとき、地図領域をある程度の領域に分割し、分割領域単位でリンクごとのフィルタを生成する。分割領域は任意の大きさに設定することが可能である。

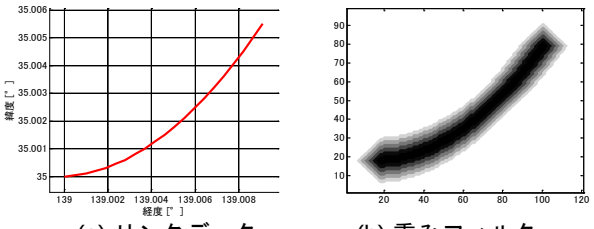
3.2 リンク別の重みフィルタ設計

提案手法で用いるリンク別の重みフィルタは、以下の手順により作成する。

＜フィルタ作成手順＞
(1) 地図領域をフィルタのサイズで領域分割する
(2) 分割領域を任意の大きさで格子分割する
(3) リンクと各格子との距離を計算する
(4) リンクとの距離やリンクの方位に応じて、各格子の重み係数を決定する
(5) 各格子の値をフィルタとして記述する

提案手法では、地図領域の格子分割は、任意の大きさに設定することが可能である。そのため、プローブデータにおける位置情報のもつ情報量を損なわずに、かつ誤差を考慮したマップマッチングを行い、データを集計することができる。

フィルタ作成の例として、図 1(a)に推定するリンクの形状を、図 1(b)に距離を重みとして作成したフィルタをそれぞれ示す。図 1(b)のフィルタでは、色の濃淡でフィルタの重み係数を表している。図 1(b)に示すように、リンクとの距離が大きくなるにつれて重みが小さくなるフィルタを生成することで、リンクからの距離を重みとして反映することが可能となる。



(a) リンクデータ　　(b) 重みフィルタ

図 1　推定リンクに対する重みフィルタ

4. 計算時間に関する見積り

ここで、既存手法および提案手法の計算時間について、理論式から比較を行う。このとき、全国におけるリンク数を N、プローブデータ数を M として、計算時間を考察する。カーナビにおいて経路案内が可能な幅 5.5[m]以上の道路が約 83 万[km]であり、これをリンク数に換算するとリンク数 N は数十~百万程度が想定される。また、プローブデータ数 M は今後の普及も加味すると毎秒数十万程度の処理を想定する必要がある。

4.1 既存手法における計算時間

既存手法では、地図領域を格子分割してプローブデータを格子毎に分類することにより、距離計算回数を削減する。このとき、格子毎のリンク平均数を n_1、プローブデータ平均数を m_1 とする。格子分割数が K_1 であると仮定すると、$N = n_1 \times K_1$、$M = m_1 \times K_1$ である。

ここで、リンク 1 本とプローブデータ 1 点の距離計算に要する計算時間を考える。リンク形状を考慮して距離計算を行うため、ここでは、リンクを構成する各ノードに対し、隣接ノード間を結んだ各直線それぞれとの距離を計算し、距離最小のものをリンクとプローブデータとの距離とする。このとき、隣接ノード間を結んだ直線とプローブデータとの距離計算を t_1 とすると、リンク 1 本とプローブデータ 1 点の距離計算に要する計算時間 C_i は式(1)のようになる。

$$C_i = (l_i - 1) \cdot t_1 \quad \ldots(1)$$

式(1)において、l_i をリンク 1 本あたりのノード数とするより、格子毎のマップマッチングに要する計算時間 T_{11} は、式(2)の通りとなる。また、全プローブデータに対する総計算時間 T_1 について、プローブデータが属する格子領域を決定する時間 $T_{12}=M \times t_0$ を加味すると、式(3)の通りとなる。

$$T_{11} = m_1 \sum_i^{n_1} C_i = m_1 \sum_i^{n_1} (l_i - 1) \cdot t_1 \quad \ldots(2)$$

$$\begin{aligned} T_1 &= K_1 \cdot T_{11} + T_{12} \\ &= K_1 \cdot m_1 \sum_i^{n_1} (l_i - 1) \cdot t_1 + M \cdot t_0 \\ &= M \cdot \left(\sum_i^{n_1} (l_i - 1) \cdot t_1 + t_0 \right) \quad \ldots(3) \end{aligned}$$

ここで、リンク数 n_1 は高々数リンクであることから無視すると、式(3)より、既存手法によるマップマッチングの計算時間は、プローブデータ数 M、リンクの構成ノード数 l_i に依存することが分かった。

4.2 提案手法における計算時間

提案手法では、フィルタ計算によるマップマッチングを行うことにより、計算時間の削減を図る。フィルタ適用のための分割領域毎のリンク平均数を n_2、プローブデータ平均数を m_2 とする。分割領域数が K_2 であると仮定すると、$N=n_2 \times K_2$、$M=m_2 \times K_2$ である。

ここで、リンク 1 本とプローブデータ 1 点のマッチングに要する計算時間を C_j とすると、フィルタ計算に要する計算時間 t_2 と等しい($C_j = t_2$)。そのため、分割領域のマップマッチングに要する計算時間 T_{21} は式(4)の通りとなる。また、全プローブデータに対する総計算時間 T_2 について、プローブデータが属する分割領域を決定する時間 $T_{22}=M \times t_0$ を加味すると式(5)の通りとなる。

$$T_{21} = m_2 \sum_j^{n_2} C_j = m_2 \cdot n_2 \cdot t_2 \quad \ldots(4)$$

$$\begin{aligned} T_2 &= K_2 \cdot T_{21} + T_{22} \\ &= K_2 \cdot m_2 \cdot n_2 \cdot t_2 + M \cdot t_0 \\ &= M \cdot (n_2 \cdot t_2 + t_0) \quad \ldots(5) \end{aligned}$$

ここで、リンク数 n_2 は分割領域を決定してしまえば定数となるため無視すると、式(5)より、提案手法によるマップマッチングの計算時間は、プローブデータ数 M にのみ依存することが分かった。

このとき、提案手法の既存手法に対する計算速度の割合を R_{12} とすると、R_{12} は式(6)の通りとなる。

$$R_{12} = T_1 / T_2 = \frac{M \cdot \left(\sum_i^{n_1} (l_i - 1) \cdot t_1 + t_0 \right)}{M \cdot (n_2 \cdot t_2 + t_0)} = \frac{\sum_i^{n_1} (l_i - 1) \cdot t_1 + t_0}{n_2 \cdot t_2 + t_0} \quad ...(6)$$

式(6)より、計算速度の向上はマップマッチングの対象となるリンクの数とともに、リンクの構成ノード数によって影響が出ることが分かった。

以上から、領域内のリンク数とリンク構成ノード数によって、それぞれの計算時間が変化することが分かった。提案手法において距離計算をフィルタ計算に置き換えたことで、リンクのノード数に依存しない計算法であることが分かった。これにより、特にリンクの構成ノード数が多い場合に、リンクのノード数分の計算時間改善効果が期待できる。プローブデータ数 M は今後の普及拡大に伴う大規模化が想定され、プローブデータ数以外の部分での計算時間の短縮が処理の効率化に重要である。そのため、提案手法における計算時間削減の効果が大きくなると考えられる。

5. 模擬データによる計算時間の評価実験

提案手法に関して、模擬データを用いて評価実験を行った。実験は、一本のリンクに対してマップマッチングとリンク旅行時間の推定を行い、計算時間を測定した。構成ノード数の異なる複数リンクに対して、それぞれ計算時間を測定することにより、提案手法を用いた場合と既存手法を用いた場合を比較して、提案手法の効果について評価した。

5.1 提案手法を利用したリンク旅行時間の推定方法

提案手法を利用してリンク旅行時間の推定を行う場合、フィルタの重み係数を利用した重みづけ加算によって、プローブデータからリンク通過速度を推定し、これをもとにリンク旅行時間を推定する。リンク通過速度の推定は、式(7)に基づいて行う。ここで $A(i)$ は、フィルタより算出したプローブデータ $V_{probe}(i)$ における重み係数である。また、リンク旅行時間の推定は、リンク通過速度の推定結果とリンク長 L から式(8)のように算出する。

$$V_{estimation} = \frac{1}{\sum_i^n A(i)} \sum_i^n V_{probe}(i) \cdot A(i) \quad ...(7)$$

$$TTL_{estimation} = \frac{L}{V_{estimation}} \quad ...(8)$$

5.2 実験方法

計算時間評価は、一本のリンクについて、多数のプローブデータのマップマッチングを行い、リンク旅行時間を算出するまでの計算時間を測定した。このとき、マッチングさせるリンクとして、(1)直線リンク、(2)カーブリンク、(3)連続ヘアピンカーブリンクの、形状(構成ノード数)の異なる 3 種類を用意した。各リンクは、高速道路や主要道(直線、カーブ)、山道(連続ヘアピンカーブ)などで見られる道路形状である。推定する 3 本のリンク形状を図 2 に示す。また、本実験の実験条件について、各リンクのノード数およびプローブデータ数は表 1 の通りに設定した。このとき、計算時間は、プローブデータとのマップマッチング計算およびリンク旅行時間推定終了までとした。

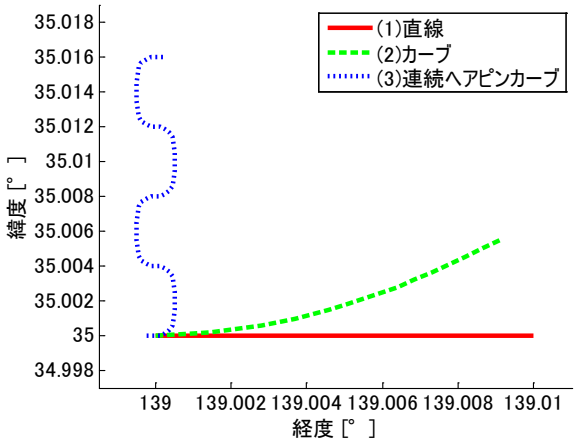


図 2　計算時間評価に用いる推定対象リンク

表 1　計算時間評価における実験条件

	ノード数 l	ノード間直線数 l-1	プローブデータ数 M
(1)直線リンク	2	1	103680
(2)カーブリンク	21	20	
(3)連続ヘアピンカーブリンク	61	60	

5.3 実験結果

上記条件において、それぞれマップマッチングからリンク旅行時間の推定を行った。計算時間の測定結果を表 2 および図 3 に示す。図 3 より、直線リンクのように構成ノード数が少ないリンクの場合は計算時間に大きな差が発生しないが、カーブが連続するリンクの様に構成ノード

数が増加した場合に、既存手法の計算時間が比例して増大しているのに対して、提案手法の計算時間がほとんど変化していないことが分かった。これより、提案手法をリンク旅行時間の推定に利用することにより、より効率的に処理することが可能となることが分かった。このときの既存手法と提案手法の計算時間増加は式(3)および式(5)とも一致する結果であった。また、リンク旅行時間の推定結果について、本実験の条件下では模擬した推定値を正確に再現していることが分かった。

表 2　提案手法および既存手法を利用したリンク旅行時間推定に要する計算時間測定結果

計算時間 [ms]	直線リンク	カーブリンク	連続ヘアピンカーブリンク
提案	204.3122	201.6001	204.1946
既存	1963.429	3594.755	8696.367
向上率	9.609945	17.83112	42.58863

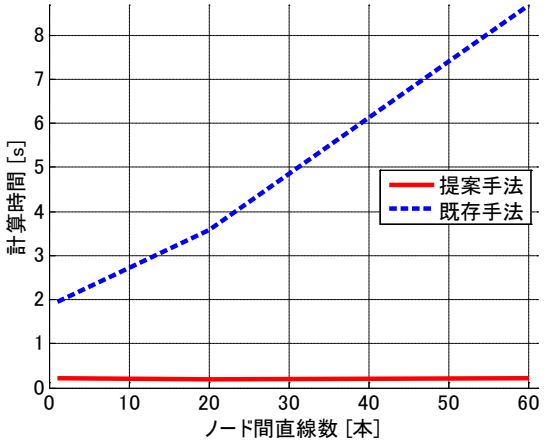


図 3　提案手法および既存手法を利用したリンク旅行時間推定に要する計算時間測定結果

6. 考察

本実験結果をもとに、一般的なリンクの構成情報を例にとって、実際に提案手法を適用した際の計算時間の向上率について考察する。地図上に構成されるリンクの多くは、主要な交差点や分岐点を端点に構成しており、国道などの主要道では 600～1,000[m]程度の単位でリンクが定義されている。一方、リンクを構成するノード点は、直線的な箇所であれば 100[m]程度、カーブのある箇所では 10～20[m]程度の間隔で設定される。これらから、1 つのリンクを構成するノード数は平均 15 点程度、最大でも 100 点程度が想定される。図 3 から、リンク 1 本に対するマップマッチングおよびリンク旅行時間推定に要する時間は、提案手法により 15 倍程度の計算速度向上を見込むことができると考えられる。

一方、マップマッチング対象となるリンク数について、文献[6]より、既存手法では、GPS 位置誤差として分散 30[m]の正規分散を想定した場合に、格子領域を 30[m]四方に設定する。この場合、マップマッチングの計算対象は 90[m]四方の領域であり、この範囲に含まれるリンク数 n_1 は 1～5 本程度であると想定される。よって既存手法によるマッチング計算は本実験結果の 1～5 倍程度になる。一方、提案手法を適用した場合を考える。分割領域は任意に設定可能であるため、分割領域を既存手法の計算対象領域と同じ 90[m]四方に設定すると、領域に含まれるリンク数 n_2 は、既存手法と同じく 1～5 本程度に収まると考えられる。よって提案手法の計算時間は 1～5 倍になる。提案手法を実際のプローブデータ処理に適用した場合について考える。また、プローブデータ数に関して、既存手法、提案手法ともにプローブデータ数の増減に依存して計算時間が変化する。以上から、本実験結果とほぼ同等の改善効果が、実際のプローブデータ処理においても期待できると考えられる。本実験では約 10 万点のプローブデータを処理する計算時間を測定した。計算時間は約 0.2[s]であったことから、提案手法は、十万単位のプローブデータを、リンクの構成によらず秒単位で処理できるマップマッチング手法であるといえる。

7. まとめ

本稿では、プローブデータの大規模化を考慮した、フィルタ演算によるマップマッチング手法を提案した。地図上のリンクは 2 つ以上のノード点により構成されており、既存手法ではノード点が多いリンクに対してマップマッチング計算に時間がかかる問題があった。これに対し、提案手法ではフィルタ計算によってマップマッチングを行うことで、リンクの構成ノード数によらず、既存手法と比較して効率的に処理することが可能となった。本稿において、リンクの構成ノード数ごとの計算時間を比較し、提案手法の計算効率を評価した。その結果、既存手法はリンクの構成ノード数に依存して計算時間が増加するのに対し、提案手法はリンクの構成ノード数によらずほぼ一定の計算時間となった。本実験結果から、提案手法を実際のプローブデータ処理に用いた場合、15 倍程度の計算速度向上を見込むことができると考えられる。

以上から、提案手法を用いることにより、プローブデータの大規模化に対しても、秒単位で十万単位のプローブデータを処理し、リンク旅行時間を推定することが可能であるといえる。

参考文献

[1]船橋 賢史, 西村 茂樹, 堀口 良太, 赤羽 弘和, 小根山 裕之, “VICS 蓄積データを用いた旅行時間短期予測手法に関する研究”, 第 27 回土木計画学研究講演集, Vol.27, CD-ROM (2003)
[2]金澤 明浩, 杵渕 哲也, 毛利 仁士, 小川 智章, 市河 研一, 荒川 研一, “決定木を利用した交通渋滞予測に関する手法”, 情報処理学会研究報告, ITS, Vol.19, pp.141-148 (2004)
[3]今井 武, 柘植 正邦, 菅原 愛子, “インターナビ・フローティングカーシステムと渋滞予測について”, 国際交通安全学会誌, Vol.31, No.1, pp.39-45 (2006)
[4]熊谷 正俊, 蛭田 智昭, 奥山 真理子, 横田 孝義, “特徴空間軌跡の追跡による動的交通状況予測”, 情報処理学会論文誌, Vol.53, No.1, pp.243-250 (2012)
[5]国土交通省 国土交通政策研究所, “プローブデータを活用した交通情報の把握に関する研究”, 次世代マルチモーダル ITS 研究会報告, pp.20-40 (2005)
[6]今井 照之, 藤山 健一郎, 喜田 弘司, 中村 暢達, “大規模プローブカー情報を処理するための高速リンクマッチング手法”, 情報処理学会第 70 回全国大会講演論文集, Vol.70, No.3, pp.37-38 (2008)