# Trajectory Segmentation Based Map-Matching Algorithm Using Least Common Subsequence (LCS) Based Scoring.

## Table of Contents

# 1. Introduction:

With the development of smartphones and portable GPS devices, largescale, high-resolution GPS data can be collected. Map matching is a critical step in studying vehicle driving activity and recognizing network traffic conditions from the data. A new **trajectory segmentation** map-matching algorithm is proposed to deal accurately and efficiently with large-scale, high-resolution GPS trajectory data. The new algorithm separated the GPS trajectory into segments. It found the shortest path for each segment in a scientific manner and ultimately generated a best-matched path for the entire trajectory. The similarity of a trajectory segment and its matched path is described by a similarity score system based on the longest common subsequence. The numerical experiment indicated that the proposed map-matching algorithm was very promising in relation to accuracy and computational efficiency. Large-scale data set applications verified that the proposed method is robust and capable of dealing with real-world, large-scale GPS data in a computationally efficient and accurate manner.

# 2. Types of Map Matching Algorithms:

1. **Incremental Algorithms:** Incremental algorithms start by selecting the matched link within an error ellipse of the trajectory points. Next, they search each following GPS point to find the best-matched path. Incremental algorithms are commonly used in mobile navigation applications because of their rapid realtime processing capability.

2. **Global Algorithms:** Global algorithms build the best-matched path by considering all trajectory points as a whole (that is, by considering the segmentwise trajectory rather than individual GPS points, as is the case for incremental algorithms). This property of global algorithms makes them well suited for large-scale GPS data processing.

# 3. Issues In Current Global Algorithms:

- **Segmentwise Trajectories Technique Limitations:**
    a. Uses constant-segment lengths, lacking flexibility.
    b. Difficulty in determining the best segmentation schema.

- **Candidate Path Set Mechanism Issues:**
    a. Challenging to maintain due to indeterminate optimal size.
    b. Needs to be large enough to ensure algorithm correctness.
    c. Must be small for computational efficiency.

- **Computational Resource Concerns:**
    a. Fixed size candidate path sets can still burden memory resources.
    b. Directly affects computational efficiency.

- **Performance and Efficiency:**
    a. Processing time ranges significantly, from about 0.2 to 2.0 seconds per point.

# 4. Remedies of Issues of Current Global Algorithms by Trajectory Segmentation Based Map-Matching Algorithm:

- **Trajectory Segmentation Strategy:**
    a. Processes large-scale, high-resolution GPS data both accurately and efficiently.
    b. Iteratively segments the trajectory until an optimal segmentation schema is achieved.

- **Optimization Model for Segmentation:**
    a. Determines the minimum number of trajectory segments required.

b. Ensures each segment's similarity score exceeds a threshold for best subpath matching.

- **Combination of Matched Subpaths:**
  a. Constructs a best-matched path for the entire trajectory from individual segment matches.

- **LCS-Based Similarity Scoring:**
  a. Utilizes the longest common subsequence (LCS) to measure similarity.
  b. LCS allows for non-consecutive matches within the original sequences, adding flexibility.

- **Curve Similarity Optimization for Segmentation:**
  a. Proposes a model focused on curve similarity to minimize the number of segments.
  b. Each segment maintains a satisfactory similarity score with the matched path.

- **Elimination of Candidate Path Set:**
  a. Removes the need for maintaining a candidate path set during map matching.
  b. Significantly enhances the computational efficiency of the algorithm.

## 5. Algorithm Methodology:

The algorithm breaks down a vehicle's trajectory into segments to determine the most accurate overall path match. It operates under the assumption that any sufficiently small segment of the vehicle trajectory will have its matched path as the shortest path between the same start and end points of the segment. If a segment does not match the shortest path, the algorithm considers that the segment may not be small enough and that there might be a better match available by further breaking down that segment into sub segments. Below are given the key points of the algorithms methodology:

1. **Small-Segment Assumption**: Small enough segments of a trajectory are presumed to match the shortest path between their start and end points.
2. **Segmentation for Accurate Matching:** The trajectory is divided into segments that are likely to follow the shortest path, balancing between too many tiny segments and no segmentation at all.

3. **Avoiding Extremes:** The algorithm avoids the extremes of processing each pair of consecutive points as separate segments (inefficient) and not segmenting at all (potentially inaccurate).

4. **Segmentation Optimization Problem:** The method seeks to minimize the number of segments while ensuring that each segment's similarity score meets a predefined threshold, thereby optimizing computational efficiency and accuracy.

# 6. Algorithm Framework:

The method begins by finding the shortest path between the cleaned motorized trip GPS trajectory start and end points. The LCS-based similarity score is used to compare the similarity of the identified path to the GPS points' trajectory. If the similarity score of the trajectory sequence is high and the identified path matches the trajectory sequence well, the algorithm stops. Otherwise, the original trajectory is divided into segments, and a trajectory segmentation scheme is initiated. At the next iteration, the trajectory with the new segmentation scheme will be fed into the recursive process, which comprises path-finding and trajectory segmentation procedures. The algorithm continues until the new segmentation scheme is identical to the previous iteration. When the algorithm stops, the final map-matched path is obtained from the segment matched paths. The definitions for the algorithm are given below:

1. **G**: Road network **G = {V, E}**, where **V** is the set of nodes and **E** is the set of links–edge.

2. **traj0**: Original trajectory with **N** GPS track points; **traj0 = {p1, p2, . . . , pN}**, where **pi = (loni, lati)** are the GPS track point coordinates.
3. **trajT:** Trajectory segment sequence scheme with **M** segments at iteration **T**; **trajT = {S1, S2, . . . , SM}**.

The main steps include initialization, a recursive process, and a stop criterion. The details are given below:

1. **Initialization:** The original trajectory traj0 and road network G are prepared, and an initial trajectory-matching procedure is conducted in this step. A shortest path is generated according to the start and end nodes of traj0. If the similarity score (defined in the section on path finding) of the shortest path and traj0 is higher than a threshold, the best-matched path is determined and the algorithm stops. Otherwise, traj0 will be divided into segments, and the recursive process is triggered.

2. **Recursive process:** The input of the trajectory segment sequence at iteration T is trajT. The recursive process includes two substeps:
    a. **Path finding**: An LCS similarity score–based path-finding method is used to find the corresponding shortest path with the best similarity score for each segment. If the segment path similarity score is greater than a threshold $\delta$ (for example, score $\geq \delta = 0.95$), the segment is seen as a similar segment. Otherwise, the segment is classified as a dissimilar segment.
    b. **Trajectory segmentation:** A trajectory segmentation method is applied to all segments either to separate each dissimilar segment into subsegments according to similarity characteristics and cutting points or to skip the similar segments. A score validation procedure is used to guarantee that the similarity score improves because of the segmentation procedure. If the score validation fails, the targeted dissimilar segment stays the same. These steps end up generating a new trajectory segmentation scheme trajT+1.

3. **Stop criterion**: The stop criterion is that the trajectory segmentation scheme does not change in consecutive iterations (i.e., trajT = trajT+1).

## 7. Algorithm's Error Tolerance Mechanism:

An additional error tolerance mechanism is applied in the algorithm for those unmatchable GPS trajectory segments. When the trajectory segment cannot find a matched path, the algorithm will skip this segment and restart on the next segment, which means the path-finding procedure will take the start and end point locations of the next available trajectory segment as the origin and destination for path searching.

## 8. Path Finding:

The procedure first finds the network end node and start node sets according to the endpoints of the network links that are spatially close to the trajectory segment end and start point locations. The start node set is usually selected as the previous end node of the segmentmatched path to keep the connectivity of the matched path except for first-segment situations or the map matching restarting from a malfunction situation. The method computes a group of candidate shortest paths from all start nodes to all end nodes using Dijkstra's shortest path algorithm. Among the candidate shortest path group, the method calculates a similarity score for each candidate shortest path and selects the path with the highest similarity score.

The similarity score describes the similarity of a GPS trajectory segment to its corresponding path. For example, assume a track point **pi = (x, y)i** from the trajectory segment **Sn = {p1, p2, . . . , pN}** and a **linkj = (xs, ys, xe, ye)j** from the path **Pm = {link1, link2, . . . , linkM}**. The spatial proximity of point **pi** to **linkj** is described by a **simobj** function. A function value greater than zero indicates that some degree of similarity is obtained.

$$\text{simobj}(p_i, \text{link}_j) = \begin{cases} 0 & \text{dist}(p_i, \text{link}_j) > \varepsilon \\ 1 - \dfrac{\text{dist}(p_i, \text{link}_j)}{\varepsilon} & \text{otherwise} \end{cases}$$

where **dist(pi, linkj)** represents the point-to-line distance between **pi** and **linkj** and **ε** is the maximal distance threshold for determining how similar two objects are. The **simobj** function returns a value between **0** and **1**. The **zero** value indicates two objects are not similar at all. A value close to **one** indicates that two objects are very similar.

A dynamic programming model is used to calculate the **LCS** score of the trajectory segment **Sn** and the path **Pm**. Let **Sn(i)** denote the first **i** points of **Sn** such that **Sn(i) = {p1, p2, . . . , pi}**, where **i ≤ N**, and let **Pm( j)** denote the first **j** links of path **Pm**, such that **Pm( j) = {link1, link2, . . . , linkj}**, where **j ≤ M**. A recursive function **scoreLCS(Sn(i), Pm( j))** is defined to obtain the LCS score for those two elements. When **i = 0 or j = 0**, **scoreLCS(Sn(i), Pm( j)) = 0**. If **i ≠ 0** and **j ≠ 0** then

$$\begin{aligned} &\text{scoreLCS}(S_n(i), P_m(j)) = \\ &\quad \max\big(\text{scoreLCS}(S_n(i-1), P_m(j-1)) + \text{simobj}(p_i, l_j), \\ &\quad \text{scoreLCS}(S_n(i-1), P_m(j)), \text{scoreLCS}(S_n(i), P_m(j-1))\big) \end{aligned}$$

The function **scoreLCS** returns a similarity score of the best match between **Sn** and **Pm** up to their **ith** and **jth** objects. The LCS score for the full sequence is thus given as **scoreLCS(Sn(N), Pm(M))**. The **scoreLCS** function value is the summation of all **simobj** values for all matched object pairs. Because the **simobj** value is between **0** and **1**, if one assumes the two sequences are exactly matched, the longest common subsequence is one of the two sequences. It means the **scoreLCS** value would not be greater than either **M** or **N**. Then, a sequence similarity score function **simseq** is as defined below:

$$\text{simseq}(S_n, P_m) = \frac{\text{scoreLCS}(S_n(N), P_m(M))}{\min(M, N)}$$

The simseq function value is a normalized value between 0 and 1. The larger value represents more similarity between the trajectory and the path.

## 9. Trajectory Segmentation:

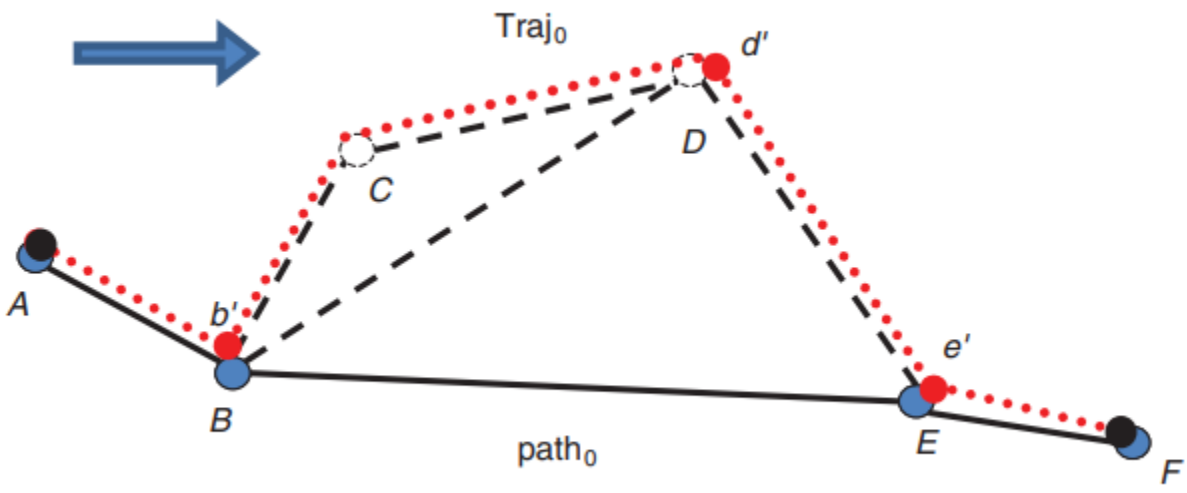
The Trajectory Segmentation part of the Algorithm optimizes map-matching by breaking down a vehicle's trajectory into smaller, more manageable segments, allowing for a more accurate matched path. The methodology is based on the principle that segments with low similarity scores are further divided using cutting points derived from the trajectory's distance profile. These cutting points are identified either as points of largest distance to the path or points near a distance threshold, indicating a deviation from the matched path.

The summarized methodology includes:

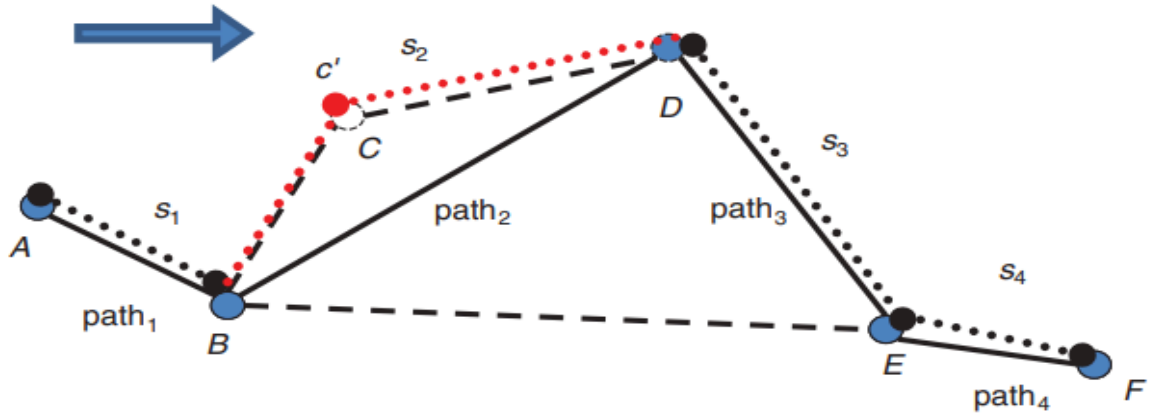
1. Dividing dissimilar trajectory segments into subsegments at cutting points based on distance profiles.
2. Finding the shortest path for each subsegment to potentially improve the match.
3. Combining all subpaths to form a new path for the trajectory.
4. Retaining the subsegment division only if the new path has a higher similarity score than the old path otherwise the dissimilar segment remains the same.
5. Maintaining similar segments as is, while updating the overall path based on new segmentation schemes.
6. Stopping the algorithm when further segmentation does not improve similarity scores.

The algorithm is demonstrated through figures showing the iterative process of segmentation and path matching, where segments are adjusted until all have satisfactory matches, at which point the algorithm concludes.

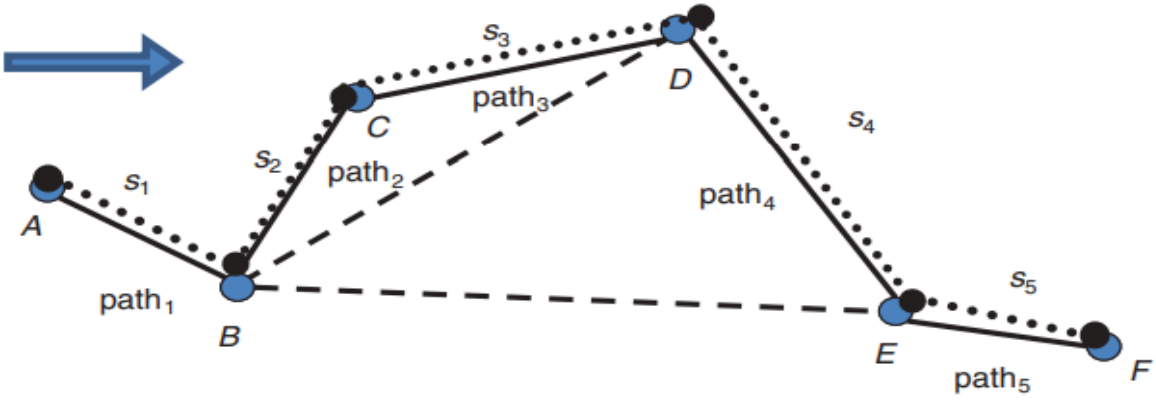
## Segmentation Algorithm Steps For Single Trip Trajectory:




(a)

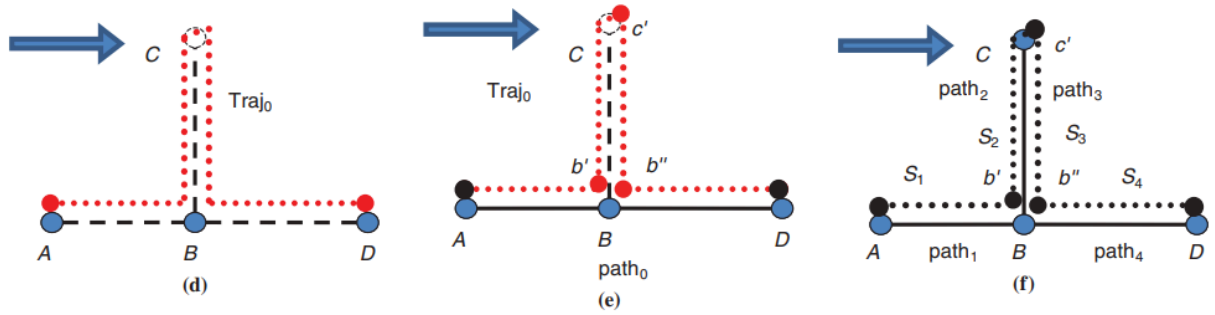



(b)

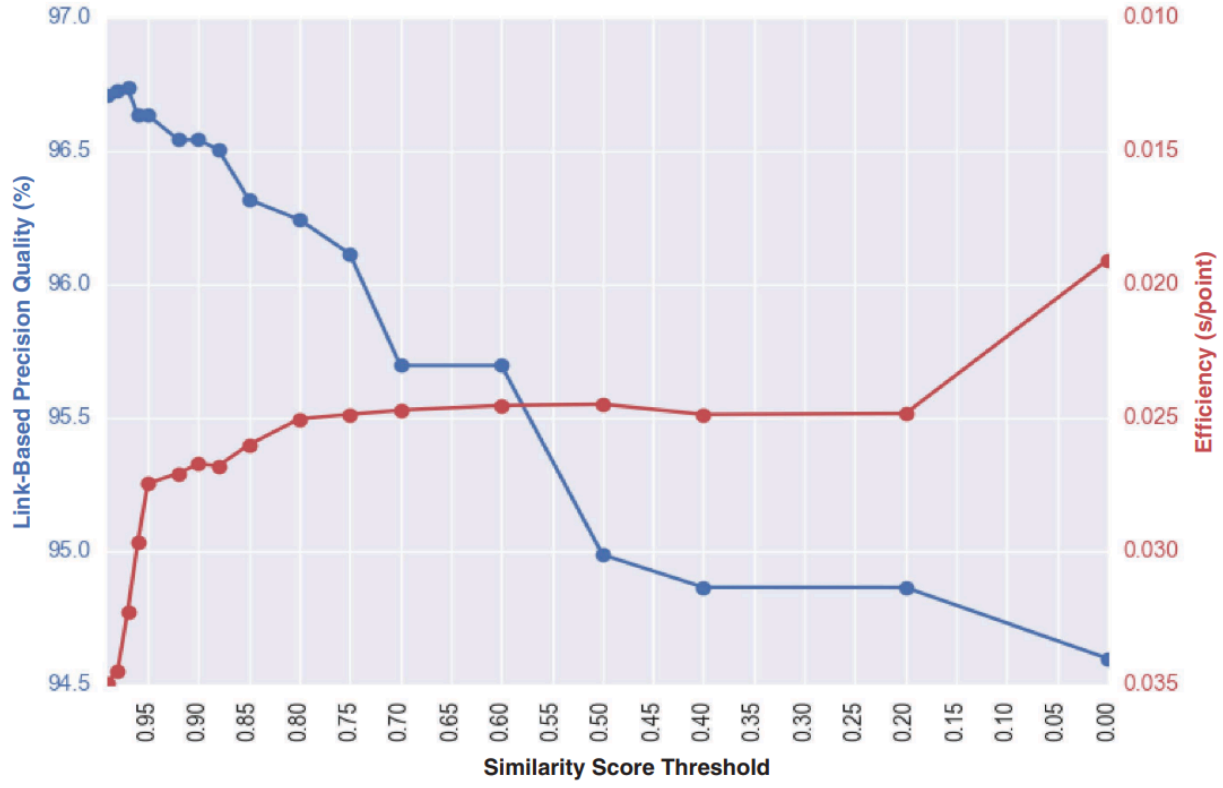



(c)

**Segmentation Algorithm Steps For Trip Chain Trajectory:**




# 10. Precision And Computational Efficiency Plot Of Algorithm:




The figure shows that as the similarity score threshold decreases, the precision quality drops and computational efficiency improves. The computational time

improves substantially with steady reductions in the similarity score threshold from 0.99 to 0.95. Further decrease in the threshold below 0.95 continues to improve the computational time but at a slower rate. At a threshold of 0.95, the computational time is 0.0275 s per point, and the precision quality is 96.6%. The results also show that average precision quality decreases as the similarity score threshold decreases, although precision quality levels out between 94.5% and 95% at threshold values for the similarity score below 0.5.

## 11. Comparison With Other Map Matching Algorithms:

The proposed map-matching algorithm was compared with other benchmarks for computational efficiency and precision quality. The **snap-to-nearest** method is one of the simplest map-matching algorithms, by which the GPS point is just snapped to the closest road network link. It is a computationally efficient algorithm but not the most accurate one. The **Hidden Markov Model (HMM)** approach to the map-matching problem is a more recent solution that provides more accurate results than the snap-to-nearest approach. The results for each of these algorithms applied to the test data set are listed in table below. The snap-to-nearest method shows the best computational efficiency, but its precision quality is much worse than that for the other methods. The results for the current LCS method are given for two similarity score thresholds to show the impact of this parameter on the comparison of computational efficiency and precision quality. Even at a similarity score threshold setting of **0%**, the LCS method shows **3%** higher precision quality and **27%** higher computational speed than the **HMM** method. Raising the similarity score threshold to **95%** results in roughly equal computational speed but a full **5%** higher precision quality for the LCS method relative to the HMM method.

| Method | Computational Time | Link-Based Precision Quality (%) |
|---|---|---|
| Snap to nearest | 0.002 s/point | 78.0 |
| HMM | 0.026 s/point | 91.5 |
| LCS, 0% | 0.019 s/point | 94.6 |
| LCS, 95% | 0.027 s/point | 96.6 |

# 12. Conclusion:

The proposed novel map-matching algorithm relies on trajectory segmentation in relation to similarity of the trajectory segment and its matched path. The basic premise of this new map-matching method is that the vehicle always follows the shortest path when the trajectory segment of interest is short enough. The similarity approach uses an LCS-based score to describe the similarity level of the trajectory segment and its matched path. The merits of the proposed map-matching algorithm are that the new method does not require a large candidate path set during the matching procedure and that the segmentation scheme is systematically generated in relation to the similarity score. With those advantages, the performance of the proposed map-matching algorithm is very promising with respect to computational efficiency and accuracy for large-scale GPS trajectory data. The link-based precision quality reaches a maximum of **96.74%** at a similarity score threshold of **0.97**. Reducing the similarity score threshold to **0.95** significantly decreases the required computational time while maintaining a precision quality of greater than **96.5%**.

# 13. References:

1. Zhu, L. Routing Map Topology Analysis and Application. PhD dissertation. University of Arizona, Tucson, 2014.

2. Zhu, L., and Y.-C. Chiu. Transportation Routing Map AbstractionApproach: Algorithm and Numerical Analysis. Transportation Research Record:

Journal of the Transportation Research Board, No. 2528, 2015, pp. 78–85. https://doi.org/10.3141/2528-09.

3. Wenk, C., R. Salas, and D. Pfoser. Addressing the Need for Map-Matching Speed: Localizing Global Curve-Matching Algorithms. In Proceedings of the 18th International Conference on Scientific and Statistical DatabaseManagement, Vienna, Austria, IEEE, 2006, pp. 379–388.

4. Perrine, K., A. Khani, and N. Ruiz-Juri. Map-Matching Algorithm for Applications in Multimodal Transportation Network Modeling. Transportation Research Record: Journal of the Transportation Research Board, No. 2537, 2015, pp. 62–70. http://dx.doi.org/10.3141/2537-07.

5. Bergroth, L., H. Hakonen, and T. Raita. A Survey of Longest Common Subsequence Algorithms. In Proceedings of the 7th International Symposium on String Processing and Information Retrieval, A Coruna, Spain, IEEE, 2000, pp. 39–48. https://doi.org/10.1109/SPIRE.2000.878178.

6. Quddus, M.A., W.Y. Ochieng, and R.B. Noland. Current Map-Matching Algorithms for Transport Applications: State-of-the Art and Future Research Directions. Transportation Research Part C: Emerging Technologies, Vol. 15, No. 5, 2007, pp. 312–328. https://doi.org/10.1016 /j.trc.2007.05.002

7. White, C.E., D. Bernstein, and A.L. Kornhauser. Some Map Matching Algorithms for Personal Navigation Assistants. Transportation Research Part C: Emerging Technologies, Vol. 8, 2000, pp. 91–108.

8. Yu, M. Improved Positioning of Land Vehicle in ITS Using Digital Map and Other Accessory Information. PhD dissertation. Hong Kong Polytechnic University, Hong Kong, 2006.

9. Quddus, M.A., W.Y. Ochieng, L. Zhao, and R.B. Noland. A General Map Matching Algorithm for Transport Telematics Applications. GPS Solutions, Vol. 7, No. 3, 2003, pp. 157–167. https://doi.org/10.1007/s10291-003-0069-z