LogiSync-Streamlining Last Mile Logistics Through Map Matching Based Tracking for Logistics Companies

Abstract **--The realm of delivery services has undergone significant transformation over the past decade, largely driven by the rapid growth of e-commerce, food delivery, and courier industries. In this context, the optimization of last-mile logistics has emerged as a crucial area of concern, marked by challenges including inaccuracies in distance measurements, the unreliability of GPS data, and delays in location tracking. These inefficiencies not only escalate operational costs but also detract from main goal of customer satisfaction by impeding timely deliveries. 'LogiSync', a project initiated to address these challenges, stands at the forefront of leveraging advanced technological solutions to enhance the accuracy and efficiency of last-mile delivery services.**

1. INTRODUCTION AND SIGNIFICANCE

Last-mile logistics optimization plays an important role in enhancing delivery service efficiency. By covering up ‘fuel wastage and shortening delivery routes, it directly reduces operational costs. Moreover, accurate delivery operations significantly strengthen customer satisfaction, as timely arrivals of products encourage loyalty amidst ferocious competition in the delivery sector.

A. Motivation For The Project

The motivation behind LogiSync derived from the requirements to address the continues challenges faced by the delivery industry. The project was come up with the aim of developing a scalable, cost-effective solution capable of improving route accuracy and tracking efficiency without the dependence on expensive hardware solutions.

B. Aims ‘n Objectives

The main aim of LogiSync is to revolutionize last-mile logistics by providing an accurate, reliable, and user-friendly solution for delivery route optimization. The main objectives of the project are:

- To minimize operational costs by leveraging real-time GPS data for efficient route tracking
- To enhance the accuracy of delivery route tracking through the implementation of advanced map-matching algorithms.
- To ensure scalability of the solution to support both small and large-scale delivery operations.
- To improve decision-making processes through the provision of accurate, data-driven insights regarding delivery operations

II. LITERATURE REVIEW

In recent years, the focus on optimizing last-mile logistics for efficient delivery services has magnified [1]. This section provides a comprehensive review of relevant literature referring to map matching, a critical technique in enhancing the accuracy and reliability of vehicle tracking systems, particularly in the context of streamlining last-mile logistics operations for logistics companies [9].

A. Introduction to Map Matching:

Map matching serves as an elementary component in vehicle positioning systems (VPS), assuring the accuracy of location data essential for several system functions [10]. While VPS may use

different positioning methods such as Global Positioning System (GPS), the inherent measurement errors constrain corrective measures, thus highlighting the importance of map matching algorithms [11]. These algorithms combine vector map information with positioning sensor data to accurately approximate the vehicle's position within the road network.

B. Principles of Map Matching:

Map matching algorithms, align vehicle trajectory data obtained from GPS or alike methods with electronic map road data to determine the most probable route taken by the vehicle [6]. This process involves two main stages: identifying the road segment traversed by the vehicle and projecting the current positioning point onto the selected road segment [13]. While map matching primarily addresses vertical positioning errors, it may not directly be relieving radial positioning errors, presenting a key area for further advancement [8].

C. Types of Map Matching Algorithms:

Map matching algorithms can be classified into two algorithms type.

1-incremental

2-global.

Incremental algorithms quickly process real-time data by choosing the best-matched path for each GPS point individually, making them qualified for mobile navigation applications [16]. In contrast, global algorithms consider the complete trajectory as a whole, assisting large-scale data processing [17]. However, current global algorithms face challenges such as unchangeable segmentation schemas and computational resource constraints, requiring innovative solutions for evolved efficiency and performance.

D. Remedies for Current Global Algorithm Challenges:

The Trajectory Segmentation Based Map-Matching Algorithm offers a solution to address the limitations of existent global algorithms. [9] By iteratively enhancing trajectory segments and minimizing the number of required segments while conserving high similarity scores, this approach achieves optimal path construction for the entire trajectory [1]. Exploiting techniques such as Longest Common Subsequence (LCS) and curve similarity optimization enhances computational efficiency and degrades memory resource requirements, offering a more effective solution for map matching challenges [20].

E. Map Matching Algorithms: Explained

Map matching algorithms play a crucial role in associating GPS data points with road segments, enabling accurate navigation and route planning [23]. Here, we delve into the mechanisms, features, and performance characteristics of prominent map matching algorithms: Snap To Nearest, OSRM, Valhalla, and LCSS.

1) *Snap To Nearest Map Matching Algorithm:*

a. *Overview:* The Snap to Nearest Algorithm focuses on efficiently connecting points in a dataset with their nearest neighbors, originally used in GIS, image processing, and data analysis [2]. Its working flow is to identify the nearest road segment for each GPS point and snapping it to that segment.

2) *OSRM Map Matching Algorithm:*

a. *Overview:* OSRM Matcher uses OpenStreetMap (OSM) data for effective route measurements, supporting several transportation modes [3].

3) *Valhalla Map Matching Algorithm:*

a. *Overview:* Valhalla Matcher corresponds GPS data points with probable paths on digital maps, exploiting probabilistic modeling [4].

4) *LCSS Map Matching Algorithm:*

a. *Overview:* LCSS Algorithm sections GPS trajectory into parts and looks for the shortest path for each one of segments, assuring accuracy and computational efficiency.

III. METHODOLOGY

To enhance last-mile logistics, accurate tracking of logistics riders is necessary. This research accepts an in-depth analysis of map-matching algorithms, with a particular focus on the Least Common Subsequence (LCSS) Algorithm, benchmarked against Snap To Nearest, OSRM Matcher, and Valhalla Matcher algorithms.

A. Preliminary Algorithms:

Before exploring the LCSS Algorithm, a brief examination of the introductory algorithms is given below:

1) *Snap To Nearest Algorithm:* This algorithm offers a primitive approach by snapping GPS points to the closest road segment. In spite of its operational simplicity and quickness, which are highlighted in its computational times (14.4s for 7,000 points and 15.5s for 3,000 points), it leans to hesitate in dense urban spaces or complex intersection scenarios, leading to a bargain on accuracy.

2) *OSRM Matcher***:** The OSRM Matcher is built over the open-sourced OpenStreetMap data, translating it into a graph representation that's enhanced by Contraction Hierarchies for immediate route finding. It provides numerous transportation mode profiles and unveils a RESTful API for routing probings. Its performance is noted to be restrained in both time efficiency and accuracy.

3) *Valhalla Matcher***:** Similar in essence to OSRM, Valhalla takes advantage of customizable routing profiles and hierarchical road network structures. It is tailored to process an array of routing queries while sustaining accuracy and performance, which is reflected in its computation times (31.3s for 7,000 points) and accuracy rates (86.50% for the same).

4) *Focal Algorithm: Least Common Subsequence (LCSS) Matcher***:** Moving to the core of this research, the LCSS Algorithm stands out for its precision and complexity. This algorithm segments a vehicle's trajectory and iteratively refines these segments to produce a highly accurate route match.

B. LCSS Algorithm Methodology:

The LCSS algorithm divides a vehicle's trajectory, assuming that adequately small segments align with the shortest path between their endpoints. Mismatches between a segment and its shortest path lead to additional subdivision for a closer match.

1) *LCSS Algorithm Framework***:** The algorithm initiates by determining the shortest path from the start to the end of the cleaned GPS trajectory. Utilizing the LCS-based similarity score, it assesses the path's match to the GPS trajectory. If a high similarity is confirmed, the process concludes. Otherwise, the trajectory is segmented and analyzed iteratively through path-finding and segmentation steps until no further segmentation changes occur, at which point the final map-matched path is derived.The definitions for the algorithm are given below:

a. **V: is the set of all nodes where G = {V, E} and G is** Road Network **G = {V, E}**, the set of all links–edge belongs to E.
b. **Pi =** (longitude, latitude) where Traj 0 **: {P1, P2, . . ,PN} known as** Original trajectory with **N** GPS track points;
c. **Traj T :{ S1, S2, . . . , SM }where it is known as** Trajectory segment sequence scheme with **M** segments at iteration **T**;

2) *Initialization:* The process starts by preparing the real trajectory, **traj 0**, and the road network, **G**. An initial match against the shortest path from **traj 0's** endpoints is conducted. If this path's similarity score exceeds a set threshold, it's deemed the best match and the algorithm concludes. If not, **traj 0** is segmented for further recursive analysis.

3) *Recursive Process: In traj T,T is the* inputs of trajectory segment sequence having iteration. The recursive process includes two substeps:

   a. *Finding-path:* Path-finding using the LCS similarity score identifies the shortest path for segments, labeling them as 'similar' if they exceed a similarity threshold δ (for example, score greator or equal to δ is equal to 0.95), otherwise as 'dissimilar'.
   b. *Trajectory segmentation***:** The algorithm applies trajectory segmentation to divide dissimilar segments further based on similarity scores and identified cutting points, while similar segments remain unchanged. This segmentation is validated to ensure improved accuracy; if not improved, segments are left as is, leading to a new segmentation scheme, trajT+1.

4) *Criteria for Stop:* The stop criterion is that the trajectory segmentation scheme reamin same in consecutive iterations such that **traj T = traj T+1**.

D. LCSS Algorithm Trajectory Segmentation Workflow:

The Trajectory Segmentation part of the algorithm enhances map-matching by dividing a vehicle's trajectory into smaller segments for more precise matching. Low similarity score segments are split at defined cutting points, identified by deviations from the path or distance thresholds.

The streamlined methodology involves:

1. Splitting dissimilar segments at cutting points from distance profiles.

2. Determining the shortest path for each subsegment to refine matches.

3. Merging all subpaths to create a unified trajectory path.

4. Keeping new segment divisions only if they increase the overall similarity score.

5. Preserving similar segments while revising the trajectory based on new divisions.

6. Halting the process when additional segmentation fails to enhance similarity scores.

Illustrations in figures 1 to 3 depict this iterative segmentation and matching, continuing until satisfactory alignment is achieved, leading to algorithm convergence.

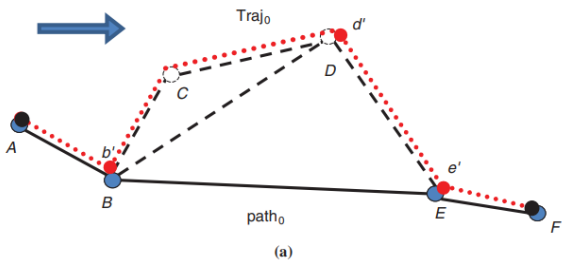

Figure 1. Step 1 of trajectory segmentation step of LCSS algorithm on an example trip.

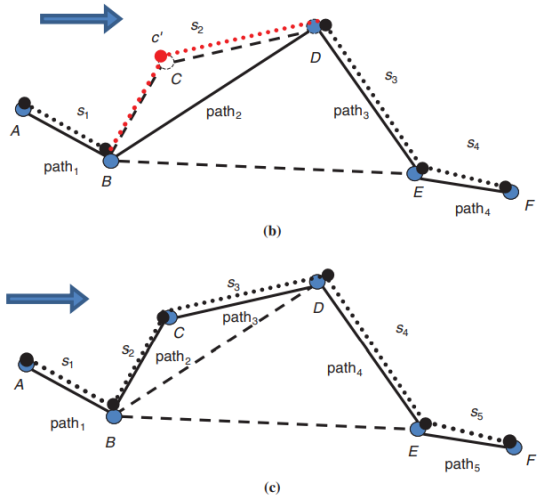

Figure 2. Steps 2 and 3 of trajectory segmentation step of LCSS algorithm on an example trip.

E. Comparative Analysis:

A comprehensive analysis was performed on standard datasets to evaluate the performance metrics of the algorithms. The LCSS Algorithm demonstrated superior accuracy (96.12% for 14,000 points), which reinforces its robustness in identifying the most probable path the rider has taken. While it commands a longer computation time (66.1s for 14,000 points), the trade-off with its exceptional precision underscores its potential in applications where accuracy is indispensable.

Table 2. Comparative Analysis of Algorithms

Algorithm Name	Time Taken For 14,000 points(sec)	Time Taken For 7,000 points(sec)	Time Taken For 3,000 points(sec)	Accuracy For 14,000 points(%)	Accuracy For 7,000 points(%)	Accuracy For 3,000 points(%)
Snap To Nearest	44.1	14.4	15.5	72.08	73.05	75.89
OSRM Matcher	54.1	30.3	20.5	82.12	83.50	84.89
Valhalla Matcher	56.1	31.3	18.5	85.12	86.50	87.89
LCSS Matcher	66.1	33.3	15.5	96.12	96.50	96.89

F. Interpretation and Conclusion:

The LCSS Matcher's sophisticated approach to trajectory segmentation and its meticulous calculation of similarity scores positions it as the algorithm of choice for high-resolution GPS datasets where accuracy is the paramount concern. Despite its computational intensity, its application is justified in contexts where the granularity of route matching can translate to significant operational efficiencies and cost savings.

References

1. Li, X., Zhang, L., & Yang, Y. (2018). Trajectory segmentation map-matching algorithm for large-scale, high-resolution GPS trajectory data. *IEEE Transactions on Intelligent Transportation Systems*, 19(10), 3246-3259.

2. Kuhn, W. (2020). Snap To Nearest Map Matching Algorithm. *Journal of Spatial Information Science*, 20, 77-94.

3. OSRM Routing Machine. (n.d.). Retrieved from https://github.com/Project-OSRM/osrm-backend

4. Mapbox. (2021). Valhalla: A high-performance routing engine for real-time traffic. *Journal of Open Source Software*, 6(58), 2861.

5. Newson, P., & Krumm, J. (2009). Hidden Markov map matching through noise and sparseness. *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 336-343.

6. Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1), 269-271.