By : Mohamed Dhia TURKI
Work Placement at Salzburg Research
June - August 2023

Matching trajectories: Identifying similar trajectories using an adapted spatiotemporal LCSS measure.

1. Objectives

With the multitude of traffic objects and systems detecting them, the identification of the same object tracked with different sensors poses some difficulties. From differences in sampling rates, to issues with the precision of measurements, many factors contribute to the complexity of identifying trajectories that originated from the same object. We propose an approach to calculate similarities between trajectories detected by different systems and matching the trajectories that originate from the same object. This can be the initial step to further studies involving comparing trajectory detection systems or enhancing the quality of traffic datasets with data fusion techniques.

2. Literature review

In this section we present a summary of trajectory similarity measures that have been used in diverse scientific communities. From trying to study animal behavior in ecology to understanding vehicle routes in mobility analysis, trajectory similarity has been investigated in many scientific fields.

a. Similarity measures

Trajectory similarity measures aim to quantify the extent to which two trajectories resemble each other. We go through a selection of similarity measures and their characteristics.

Euclidean Distance ED

Euclidean distance is the basic and widely used measure of trajectory similarity due to its simplicity and computational efficiency. It provides an excellent starting point for trajectory analysis, especially when spatial distance is the primary focus. However, its limitations, such as sensitivity to outliers and inability to capture transient information, should be considered when choosing an appropriate similarity measure for specific trajectory analysis task.

Dynamic Time Warping DTW

Dynamic Time Warping (DTW) is a classical dynamic programming algorithm used to measure the similarity between two trajectories by searching through all possible points' alignment for the one with minimal distance. It aligns a point of one trajectory to one or more consecutive points of another trajectory. This allows for non-linear time warping, adapting to changes in time intervals, making it well suited for comparing trajectories of different lengths. DTW is

sensitive to noisy data, and small noise in trajectories can lead to significant changes in the alignment, affecting the similarity measure.

Edit Distance methods

Also known as sequence alignment methods, Edit Distance methods are similarity measures used to quantify the similarity between two trajectories by finding the minimum number of edit operations (insertion, deletion, and substitution of trajectory points) required to transform one trajectory into the other.

Edit Distance on Real sequence EDR

EDR sets a parameter *cost* to measure the cost of an unmatched pair of points. EDR searches for a minimum cost path in the distance matrix. EDR costs are set to 0 if the current pair of points matches, whereas in all other situations the cost is 1, irrespective of the distance between the current pair of points. This results in the distance threshold matrix, a Boolean matrix. The performance of EDR can depend on parameter choices (e.g., the threshold), and selecting appropriate parameter values can be challenging.

Edit Distance with Real Penalty ERP

ERP combines L p-Norm and builds on the EDR principles. It requires setting a reference point (gap point) for measuring. Instead of using the number of edits which edit distance uses, ERP uses the distance of match pairs.

Longest Common Subsequence LCSS

Longest Common Subsequence measurements attempt to quantify how well two trajectories match by determining the length of the longest point sequence that they share. LCSS matches points from the different trajectories if the distance between them satisfies the set threshold. LCSS measures are closely related to Edit Distances. LCSS requires a threshold parameter for determining point proximity, and choosing an appropriate threshold can be challenging.

Fréchet Distance FD

Fréchet Distance considers both the spatial aspect and temporal order of the curves and calculates the minimum distance required for two points to traverse their respective paths simultaneously. It is particularly useful when trajectories have different lengths, varying speeds, and undergo self-intersections or loops. FD can be sensitive to differences in sampling rates between trajectories, potentially leading to biased similarity assessments.

3. Proposed approach

Due to the characteristics of our dataset (differences in sampling rates, precision of GPS…), we found the most robust method to use is an adaptation of the Longest Common Subsequence with a time penalty to take both the spatial and temporal similarity of each point of the compared trajectories. We match each c-its trajectory to its most similar LiDAR track.

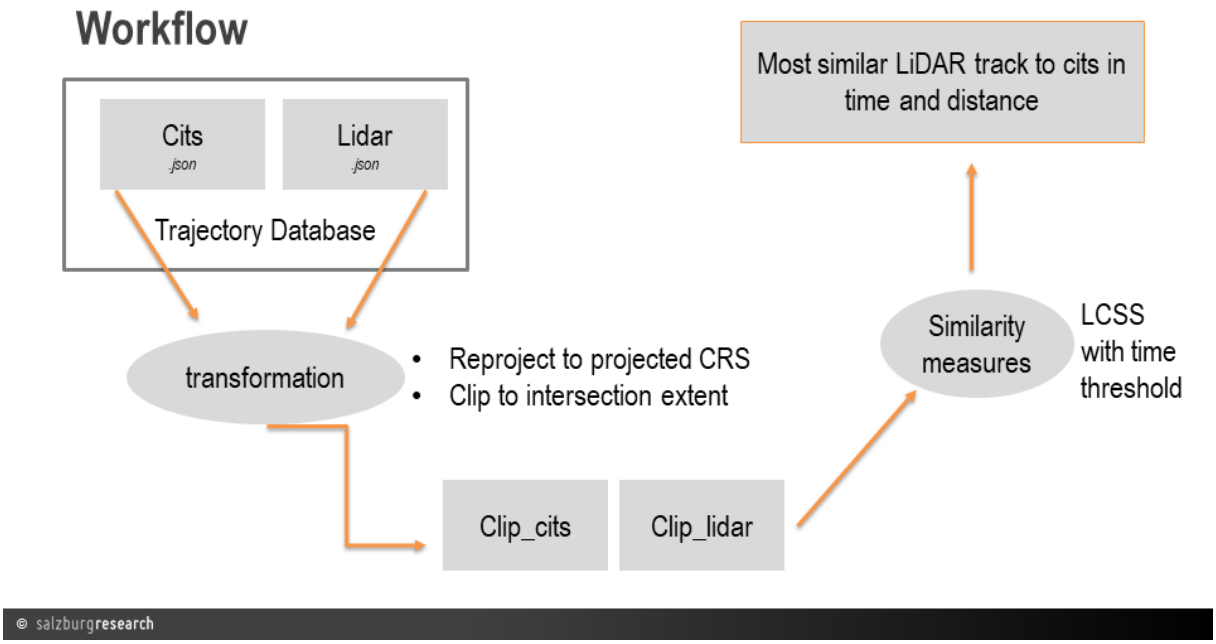
a. Dynamic programing

In dynamic programming, filling matrices involves systematically computing and storing values in a two-dimensional table. Each cell of the matrix represents the result or a particular

combination of input values. Then, we iteratively fill the remaining cells based on recursive formulas or rules, utilizing the already computed values in the table.

b. Preprocessing

Trajectories exported from the floating vehicle dataset are transformed to multiple csv files according to their track:id. This is done via a QGIS model or has also been implemented in Python.



c. LCSS-T Algorithm

We implemented the Longest Common Subsequence with Time as a dynamic programming approach to compare trajectories from two different sources (c-its and LiDAR) and find the most similar matches between them based on their spatial and temporal properties.

After proper preprocessing and data transformation, the algorithm calculates the spatial and temporal distances between each point from the compared trajectories. Three matrixes are created : one to store the length of the common subsequence, one to store the spatial distance and a last one to store the temporal distance. These matrixes are filled by means of dynamic programming.

If the spatial distance is within the pre-defined threshold and the temporal ''distance'' is within the temporal threshold, it considers those points as a potential match and fill out. The algorithm finds the longest common subsequence of such matched points, which means the longest consecutive set of points that are close both in space and time. It accumulates the spatial and temporal distances of these matches and calculates the overall similarity score, which is called LCSS-T value. The algorithm repeats this process for all possible pairs of trajectories from the c-its and LiDAR datasets, and it finally identifies the pair with the highest LCSS-T value as the best matching pair.

By using this approach, the algorithm can handle trajectories with different lengths and sampling rates, making it suitable for comparing trajectories from different sources with varying characteristics.

The LCSS-T algorithm is made available from a Jupyter Notebook, standalone Python script, and a simple GUI developed with Tkinter. The algorithm can be used to find the most similar trajectory to a reference trajectory from a pool of candidates or to pair match trajectories by similarity.