An Interactive Map-based System for Visually Exploring and Cleaning GPS Traces

Abdeltawab Hendawi
University of Rhode Island
hendawi@uri.edu

Sree Sindhu Sabbineni
University of Washington, Tacoma
ssindhu@uw.edu

Jianwei Shen
University of Washington, Tacoma
sjwjames@uw.edu

Yaxiao Song
Microsoft Corporation
yasong@microsoft.com

Peiwei Cao
Microsoft Corporation
peiweic@microsoft.com

Zhihong Zhang
Microsoft Corporation
zhz@microsoft.com

John Krumm
Microsoft Research
jckrumm@microsoft.com

Mohamed Ali
University of Washington
mhali@uw.edu

ABSTRACT

It is a fact that there are tons of GPS traces generated every minute by the millions of in-road vehicles over the world. Naturally, those traces contain imprecise readings, and most of the time they include noise and outliers. Therefore, there is a real need for a tool to allow users, companies, and researchers to get a deep insight into those raw traces and discover potential knowledge out of it. This knowledge would uncover the quality level of the GPS traces and, indeed, the quality level of the underlying map. It would also help discover interesting facts about the surrounding environment such as the type and height of buildings, the landscape cover, the weather conditions, and the nature of businesses and activities. This demo presents a system that allows users to interactively explore their collected GPS traces. Users can visually inspect the precision of their raw GPS traces, and snap these traces to the underlying road network map. Furthermore, users have the ability to clean their traces by applying various types of spatio-temporal filters. Users can perform noise analysis and produce statistics over regions of interest on the map. Last but not least, the system gives suggestions or guesses on the surrounding environment by comparing the perceived noise patterns to a database of pre-stored noise patterns. For the demo purpose, the system is initially populated with a rich data set of trajectories generated from the Microsoft shuttle service around the Greater Area of Seattle.

CCS CONCEPTS

• **Information systems** → **Spatial-temporal systems**; **Location based services**;

KEYWORDS

Data Cleaning, GPS Traces, Trajectories, Map Visualization

SIGSPATIAL '19, November 5–8, 2019, Chicago, IL, USA

ACM ISBN 978-1-4503-6909-1/19/11.
https://doi.org/10.1145/3347146.3359105

ACM Reference Format:
Abdeltawab Hendawi, Sree Sindhu Sabbineni, Jianwei Shen, Yaxiao Song, Peiwei Cao, Zhihong Zhang, John Krumm, and Mohamed Ali. 2019. An Interactive Map-based System for Visually Exploring and Cleaning GPS Traces. In *27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '19), November 5–8, 2019, Chicago, IL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3347146.3359105

1 INTRODUCTION

At the time of writing this paper, OpenStreetMap [3] has recorded that there are more than seven billion GPS points uploaded by just around five million users [11]. This number of users is less than 1% of the current 2.7 billion GPS-enabled smart phone owners over the world [12]. Other than these massive numbers of GPS readings in OpenStreetMap, there are billions of GPS points that are being collected by other mapping companies such as Bing Maps, Google Maps, MapQuest, TomTom, Garmin, and many others. It is a treasure of geo data that could lead to new knowledge and novel discoveries in location based services in general [4] such as path inference and recommendations [5–7], if leveraged efficiently. However, raw GPS traces come with noise, outliers, and inaccurate readings due to several reasons such as environmental barriers and weather conditions. Intuitively, if these raw GPS traces are used as is, this would lead to false conclusions, e.g., incorrect location of a user. Therefore, it is a common practice and a required step to clean the GPS data before actual processing. We notice that location based services begin with wiping the noise from the raw GPS readings by applying various types of filters including speed, distance, particle, and Kalman filters [8, 9, 13], then, snap (map-match) [1, 2, 10] the cleaned GPS points to the corresponding edges in the underlying road map.

In attempt to address this core need, a map-based system is demonstrated in this paper. This interactive system allows its users to visually explore their acquired GPS traces. It also allows the users to apply a set of fundamental cleaning and preprocessing operations against the GPS traces.

The main components of the demonstrated framework are: (1) *Data Loading* - Load the GPS data of the vehicle and store it to the database. (2) *Data Cleaning* - Clean the raw GPS data based

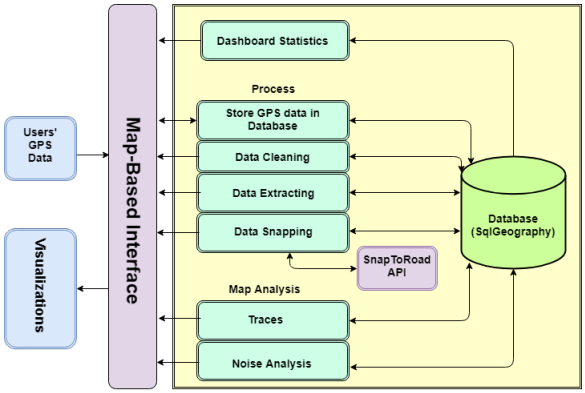

Figure 1: The Architecture of The Proposed System

on various filters such as reachability speed and location of two consecutive points. (3) *Data Extracting* - Split the raw data of the vehicle into multiple trajectories which we call as 'Runs' of a vehicle and assign a unique ID for each 'Run'. (4) *Data Snapping* - Run the 'SnapToRoad' API to obtain the snapped route for each Run of the vehicle. (5) *Visualize* - Show the snapped route and the corresponding raw data points on Bing Maps. (6) *Noise Analysis* - Produce the statistics of different regions identified based on various parameters such as tall buildings, trees, highly populated areas, parking lots etc. Initially, these statistics are generated from the Microsoft Shuttle Services data, however, it will reflect the actual statistics for the users' uploaded data.

2 SYSTEM ARCHITECTURE

Figure 1 provides the high-level description of the proposed system's architecture. It mainly consists of three modules: Dashboard, Process, and Map Analysis. The GPS data is collected from vehicles to feed into the system. The computation result would be sent to the visualizer for the display after the data is being processed.

2.1 Dashboard Module

The dashboard module displays the statistics of the loaded GPS data, (initially started with Microsoft Shuttle Services data), that is processed and stored in the database. The available GPS data is fed to the system and is processed to obtain the snapped route for each trajectory. The snapped routes are stored as SqlGeography objects in the SQL server along with the raw GPS data. Other information such as the speed of the vehicle, Heading, Timestamp etc. which we get as a response from 'SnapToRoad' API is also stored. The Dashboard module performs several computations on the obtained snapped route to analyze the patterns such as the average and standard deviation of the distance between the snapped point and the raw data point, sampling grade of the points, number of points on each day of the week, number of points for each hour of the day etc., and is sent to the front-end interface to display the statistics through textual information and interactive charts (such as histograms and bar graphs) on the dashboard tab.

2.2 Process Module

The Process module is the core functionality of the presented framework. The 'Data Loading' function of this module takes the GPS data from the user and stores it into the database as SqlGeography objects. This function also sends the stored data to the visualizer to display. The 'Data Cleaning' function filters out the data points based on the filter criterion selected by the user on the front-end interface. If the 'Filter Duplicates' filter is selected, all the duplicate points are removed from the raw GPS points and the filtered data is stored to the SQL server as SqlGeography objects. If the 'Filter by Reachability Speed' is selected, the speed required to travel from one point to the next point is calculated for every two consecutive points, and if the speed exceeds a certain threshold value, the next point is filtered out from the dataset. Similar to the 'Data Loading' function, the stored data is sent to the visualizer to display. The 'Data Extracting' functions splits the GPS points into multiple trajectories (Runs) and assign a unique ID for each Run. The user interacts with the system to specify the criterion to split the trajectories such as the maximum inter-arrival time in seconds between two consecutive points and maximum inter-arrival distance in meters between two consecutive points. The values given by the user are passed to the system from the front-end interface. Accordingly, the system performs the operation by splitting the GPS points into multiple trajectories and stores them in a table in SQL Server. The obtained trajectories are sent to the visualizer to display to the user along with the assigned RunID for each trajectory. The 'Data Snapping' function takes the RunIDs selected by the user on the interface and retrieves the GPS points for each Run from the SQL Server database. The retrieved GPS points for each Run is sent to the 'SnapToRoad' API request and the required values such as the Snapped latitude, snapped longitude, speed, timestamp, heading etc. are stored to the database table. The same information is displayed on the visualizer to the user.

2.3 Map Analysis Module

This module helps the user in understanding the data patterns of GPS devices. The 'Traces' function in this module takes the VehicleID and range of time or RunID from the user interface and retrieves the data from the database. These SqlGeography objects retrieved from the database are sent back to the visualizer to display the snapped points along with the raw data points. The 'Noise Analysis' function takes the filter criterion from the user interface and retrieves the information of polygons to display on the visualizer. Upon hover over of the displayed polygon, a request is sent to this function from the user interface to display the statistics. This function calculates several statistics in the selected region such as the number of samples, percentage of points on the left of the snapped route, percentage of points on the right of the snapped route, percentage of points matching the snapped route, distance between the snapped point and the raw data point etc. and send these computed results back to the visualizer for display.

3 DEMO SCENARIOS

In this section, we cover various scenarios for demonstrating the framework. The interactive GUI as shown in the Figure 2 is the entry point to the user. The web API is implemented in C# in Visual Studio 2019 and running on Windows 10. The front-end map-based interface is built using a combination of JavaScript, HTML, CSS and ReactJS along with the Bing Maps Interactive SDK V8. The

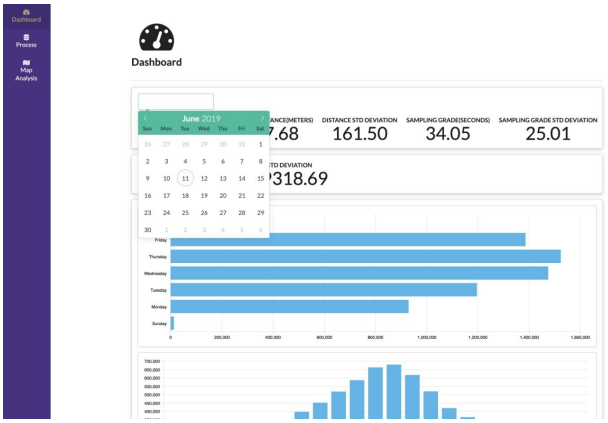

Figure 2: GUI Entry point

trajectories to examine the framework are obtained from Microsoft Shuttle Services for the months of January, February and March 2019 for the various shuttles that ran across the greater Seattle area in Washington. Demonstrated scenarios are briefly explained as follows:

3.1 Scenario 1: Dashboard

This is the entry point for the user when the GUI is launched. Dashboard displays the summarized results of the process performed on the dataset from Microsoft Shuttle Services. It gives information of various analysis through interactive charts and graphs such as bar charts and histograms along with the textual information displayed on the header. We can also check the summarized information in a certain range of time by selecting a range of dates in the calendar as shown in Figure 2. There is also an option to navigate to different tabs of the interface from side panel. The core functionality of the framework is in the 'Process' tab and the results of the analysis are displayed in the 'Map Analysis' tab.

3.2 Scenario 2: Data Loading

Users can interact with the system by loading their GPS points through the 'Data Loading' module in the 'Process' tab. Uploaded GPS data files are in Comma Separated Values (CSV) Format to store in the database. The back-end database is implemented inside Microsoft SQL-Server. Once the data is loaded, we can view the stored data along with the success message.

3.3 Scenario 3: Data Cleaning

As the raw GPS data is noisy, it is necessary to clean it before passing as input for any map-matching algorithm. In the framework, we apply filters based on the reachability speed, i.e. the speed required for the vehicle to reach the next point for every two consecutive points, and filter the points accordingly and remove any duplicates. Users can select their desired criterion from the drop down and the GPS data will be filtered out as per the selected criterion. Simultaneously, a progress bar indicates the percentage of cleaning task completion. The cleaned data is displayed along with the raw data once the data cleaning is complete as shown is Figure 3.

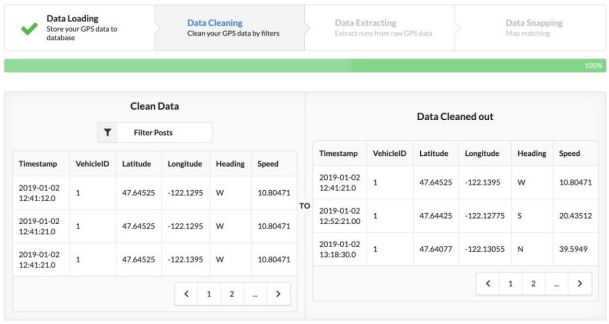

Figure 3: Listing of Cleaned Data and Filtered Out Data

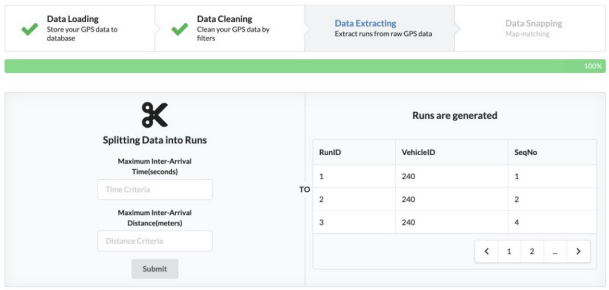

Figure 4: Splitting GPS Traces Into Runs Representing Single Trips

3.4 Scenario 4: Data Extracting

In this scenario, we split the raw GPS data into multiple trajectories which we call 'Runs' and assign a unique ID for each trajectory (Run). Each trajectory is believed to represent the GPS points for one trip of the desired shuttle. The splitting of data is based on 'maximum inter-arrival time (seconds)' and 'maximum inter-arrival distance (meters)' between two consecutive points of the data. Users have the ability to set the values for splitting the data and the system will display the details of the runs generated once the extraction of runs from the raw GPS data is complete, Figure 4.

3.5 Scenario 5: Data Snapping

The audience in this scenario can select the RunID from the extracted Runs in the data extraction scenario and submit a request to snap it to the road map. Once selected and on submission, a request is sent to the 'SnapToRoad' API. The 'SnapToRoad' API is a Microsoft Bing Maps API[1], that implements the map-matching algorithm which snaps the GPS data points to the corresponding road segment in the underlying road network graph. Several intermediate points are added to the snapped route along with the snapped points for the corresponding raw GPS points for map-matching accuracy. We can differentiate these intermediate points as they have the index value of -1 unlike the snapped points for the equivalent GPS points. The details of the snapped routes are displayed once the snapping is complete for the selected runs, Figure 5.

3.6 Scenario 6: Visualizing the Snapped Routes

The visualization of the framework on the Bing Maps is quite helpful for users to visually inspect the map-matching result for the given

[1]Bing Map Snap To Road API: https://www.microsoft.com/en-us/maps/snap-to-road.

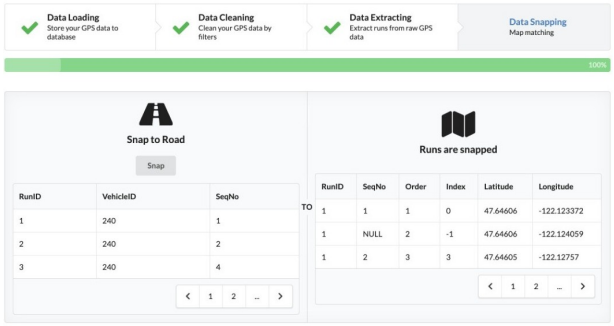

Figure 5: Snapping (Map-matching) of GPS Traces of Specific Vehicle

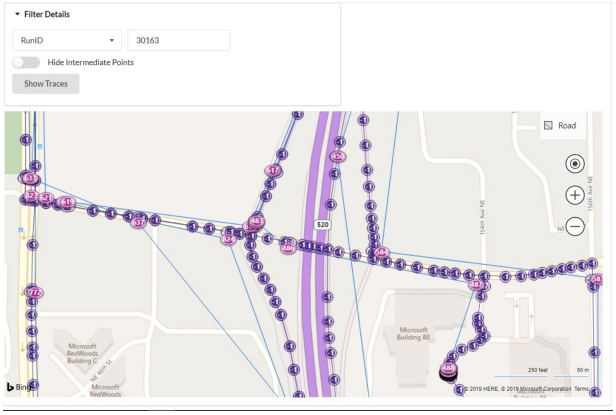

Figure 6: Visualization of Traces Based on RunID

set of GPS data. This can be accessed under 'Traces' module in 'Map Analysis' tab in the left hand-side panel. The visualization as shown in Figure 6 offers following services: (1) Users can select a specific VehicleID or RunID to visualize the results either per vehicle or per each Run of the vehicle. (2) If the VehicleID is selected, users can select the range of time to visualize the traces of that vehicle over the selected range of time. Selection of time range is mandatory as there are large number of points in the dataset for each VehicleID. (3) If the RunID is selected, we get all the points for the corresponding 'Run' of the vehicle. (4) The term 'traces' indicates the cleaned GPS points along with the snapped route and the equivalent snapped GPS points for each cleaned raw GPS point. The cleaned raw GPS points can be visualized in purple while the snapped points are shown in blue. The snapped route is displayed in blue whereas the route joining the initial cleaned raw GPS points is displayed in pale blue. (5) Recall that as mentioned in Scenario 5, there are several intermediate points between two snapped points for raw data points. These intermediate points can be visualized by turning off 'Hide Intermediate Points' button. However, the default setting is to hide the intermediate points as they are not critical for our analysis. From this visualization, users can understand the patterns of snapping at various locations of the selected runs and it also gives an estimate on the distance between the snapped point and the initial GPS point.

3.7 Scenario 7: Noise Analysis

The 'District Noise' module in 'Map Analysis' tab gives information on the general trends of the GPS devices at various regions like high

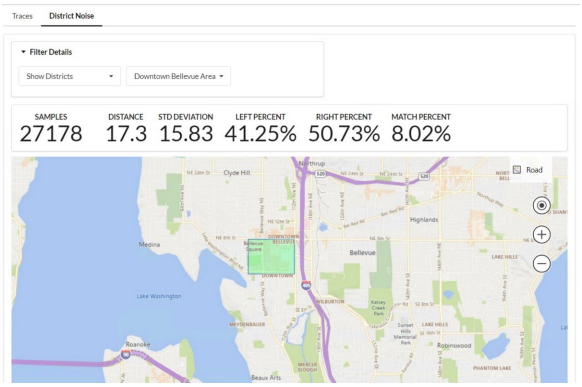

Figure 7: Display Statistics Over The Highlighted Region

Buildings, trees, parking lots etc. Users can select the desired filter criterion and the regions matching the selected filter are highlighted. By hovering over on the highlighted region, we can get the statistics for the region which are generated based on the results obtained from the Microsoft Shuttle Services data, Figure 7. Currently, these statistics provide an insight to the user about the different factors that affect the GPS device. In future, we can extend the framework to compare the results of the selected RunID/VehicleID with the existing statistics and offer reasoning behind the GPS data patterns at different locations.

REFERENCES

[1] H. Aly, A. Basalamah, and M. Youssef. Map++: A crowd-sensing system for automatic map semantics identification. In *2014 Eleventh Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 546–554. IEEE, 2014.

[2] H. Aly and M. Youssef. semmatch: Road semantics-based accurate map matching for challenging positioning data. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 5. ACM, 2015.

[3] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[4] A. M. Hendawi, M. Khalefa, H. Liu, M. Ali, and J. A. Stankovic. A vision for micro and macro location aware services. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 12. ACM, 2016.

[5] A. M. Hendawi, A. Rustum, A. A. Ahmadain, D. Hazel, A. Teredesai, D. Oliver, M. Ali, and J. A. Stankovic. Smart personalized routing for smart cities. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 1295–1306. IEEE, 2017.

[6] A. M. Hendawi, A. Rustum, A. A. Ahmadain, D. Oliver, D. Hazel, A. Teredesai, and M. Ali. Dynamic and personalized routing in prego. In *2016 17th IEEE International Conference on Mobile Data Management (MDM)*, volume 1, pages 357–360. IEEE, 2016.

[7] A. M. Hendawi, E. Sturm, D. Oliver, and S. Shekhar. Crowdpath: a framework for next generation routing services using volunteered geographic information. In *International Symposium on Spatial and Temporal Databases*, pages 456–461. Springer, 2013.

[8] W.-C. Lee and J. Krumm. Trajectory preprocessing. In *Computing with spatial trajectories*, pages 3–33. Springer, 2011.

[9] X. Liu, F. Chen, and C.-T. Lu. On detecting spatial categorical outliers. *GeoInformatica*, 18(3):501–536, 2014.

[10] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 336–343. ACM, 2009.

[11] OpenStreetMap. https://www.openstreetmap.org/stats/data_stats.html, accessed on june 11 2019.

[12] Statista. Number of smartphone users worldwide. https://www.statista.com/statistics/330695/\protect\discretionary{\char\hyphenchar\font}{}{}number-of-smartphone-users-worldwide/. accessed June 11 2018.

[13] L. Tang, X. Yang, Z. Kan, and Q. Li. Lane-level road information mining from vehicle gps trajectories based on naïve bayesian classification. *ISPRS International Journal of Geo-Information*, 4(4):2660–2680, 2015.