A Framework for Interactive Geospatial Map Cleaning using GPS Trajectories

Nikhil Vementala
Arizona State University
Tempe, Arizona
nvementa@asu.edu

Paolo Papotti
Eurecom
Biot, France
papotti@eurecom.fr

Mohamed Sarwat
Arizona State University
Tempe, Arizona
msarwat@asu.edu

ABSTRACT

A volunteered geographic information system, e.g., OpenStreetMap (OSM), collects data from volunteers to generate geospatial maps. To keep the map consistent, volunteers are expected to perform the tedious task of updating the underlying geospatial data at regular intervals. Such map curation step takes time and considerable human effort. In this paper, we propose a framework that improves the process of updating geospatial maps by automatically identifying road changes from user generated GPS traces. Since GPS traces can be sparse and noisy, the proposed framework validates the map changes with the users before propagating them to a publishable version of the map. The proposed framework achieves up to four times faster map matching performance than the state-of-the-art algorithms with only 0.1-0.3% accuracy loss.

CCS CONCEPTS

• **Database Applications, Spatial Databases and GIS**;

KEYWORDS

Map Cleaning, Map Matching Algorithm

ACM Reference format:
Nikhil Vementala, Paolo Papotti, and Mohamed Sarwat. 2017. A Framework for Interactive Geospatial Map Cleaning using GPS Trajectories. In *Proceedings of IWCTS'17:10th ACM SIGSPATIAL Workshop on Computational Transportation Science , Redondo Beach, CA, USA, November 7–10, 2017 (IWCTS'17),* 5 pages.
https://doi.org/10.1145/3151547.3151551

1 INTRODUCTION

In the past decade, digital maps have gained popularity. However, a map is useful only if it is accurate and includes the latest information. Different techniques such as Satellite Imagery and manually curated maps are employed to keep the maps updated. This process involves a lot of computing power accompanied with skilled human supervision. Due to commercial demand, coverage of places like cities on the map is higher compared to remote places.

A volunteered geographic information system, e.g., OpenStreetMap (OSM)[16], collects data from volunteers to generate geospatial maps. Information is collected using various techniques including the use of satellite/aerial imagery and the user produced GPS traces. To keep the map consistent, volunteers are expected to update the underlying geospatial data at regular intervals. Such map curation step takes time and considerable human effort. There are few third-party services that attempt to clean the OSM data in an interactive fashion. For instance, MapRoulette [14] and Kort Game[1] gamify the process of OSM bug fixing. MapCraft [10] divides the map into small pieces and enables real-time collaboration to edit maps. AddressHunter [1] improves the address coverage of the places on the map. KeepRight [13] detects errors like non-closed areas, layer conflicts, and missing name tags.

Despite these efforts, the problem of automatic detection of new roads is still to be addressed and has not received much attention. To tackle this problem, we exploit the user uploaded GPS traces and automate the process of detecting changes to the road network. As the GPS traces are sparse and possibly with imprecise measurements, the system asks the user to validate the changes before applying them to the map.

The main contributions of the paper are as follows: (i) a framework to automate the process of adding roads to the map, (ii) a new map matching algorithm to identify unmapped parts of the road network, and (iii) a module that brings the user in the loop to validate the proposed changes. The rest of the paper is organized as follows. In Section 2, we discuss the existing methods for map matching, trace aggregation, and data cleaning. In Section 3, we discuss the proposed framework, followed by experimental results on real-data in Section 4 and a discussion of the framework's limitations in Section 5.

2 RELATED WORK

Map Matching and Generation: Map Matching algorithms are used to map a given location data to a spatial road network. They are classified into four groups based on the techniques used to match a given GPS trace to the road [12]: geometric, topological, probabilistic and other advanced algorithms. Geometric algorithms are based only on the match of the closest road segment for a given point. However, GPS data can be noisy because of a weak GPS signal or low-grade equipment. Since data is prone to noise, a geometric method may lead to erroneous results, as it does not consider the continuity of the road network [12]. Topological algorithms consider the underlying structure and continuity of the road network [12]. "Probabilistic algorithms use an 'error region', which

[1]http://www.kort.ch/

IWCTS'17, November 7–10, 2017, Redondo Beach, CA, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5491-2/17/11…$15.00
https://doi.org/10.1145/3151547.3151551

is usually an ellipse or a rectangle, to match a given point" [12]. Also in this case, noise in the GPS data makes it difficult for the algorithms to match the point to the correct nearest road on the network.

Map matching algorithms based on Hidden Markov Models (HMMs) [9, 11] have set a benchmark for high accuracy and are more robust to noisy data compared to other methods. The problem of map matching also suits the HMM model well. The GPS points of the trajectory are the observations in the model and the probable roads are the states. Emission probability gives us the probability of the road to be matched to the given point. Transition probability is higher for the roads that are continuous rather than the one which are away from each other. One of the drawbacks of the HMM model is that calculating transition probability between roads requires computing the shortest path between the two roads, which is a resource intensive task. Techniques using multi-core CPUs [15], pre-calculating the transition probabilities, and considering the node distance instead of actual distance [9] can be used to speed up the process but they still need to preprocess the data to implement the solution.

The problem of autonomous map generation from scratch is addressed by [5, 7]. In this kind of work, a map is constructed from the GPS data obtained from vehicles. The data is filtered using different techniques and maps are generated after this analysis. The generated maps may not be accurate, because of inherent noise present in the data. There is need of bringing in an expert or an oracle who has knowledge of roads who can edit, or adjust proposed roads to represent the actual roads.

Trace Aggregation: Traces are the individual paths or geometrical lines on a plane. Trace aggregation is the process of replacing similar traces on the place with one representational line. Previous work proposed a solution of trace aggregation in two different ways. One approach [8] considers the LineStrings as lines and aggregates them in a 2D geometrical place. Properties like deflection of a line and the distance between adjacent lines are considered while aggregating. Algorithm in [4] is based on an image processing approach, i.e., it first converts the outliers of a cluster into an image, then applies image processing algorithms (e.g., dilation and erosion) to find the skeleton of the image and converts them back to the geometrical plane.

Data Cleaning: Work on data cleaning has focused on relational data by exploiting approaches spanning from learning to logic-based methods [2]. Popular systems rely on user-defined rules, that model patterns and constraints that must apply on the data [6]. If a set of values violate a rule, it determines that there must be an error in the data, and repair algorithms try to automatically identify it and fix it. However, such rule-based approaches do not directly apply to spatial data, This is especially true for the problem tackled in this paper, since new roads cannot be inferred from existing ones, and road closures cannot be determined by an existing topology.

3 PROPOSED FRAMEWORK

In this section, we discuss the workflow and key components of this framework. Figure 1 shows its three major components. First, the map matching algorithm which takes the GPS trajectory as an input and returns the mapped and unmapped parts of the trajectory.

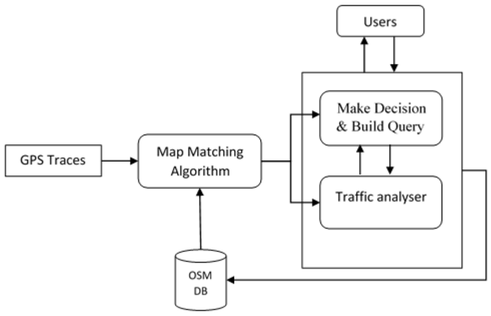

Figure 1: Architecture of proposed framework

Second, the trace analyzer which analyzes the unmapped traces of different trajectories returned by Map Matching algorithm and clusters them based on their similarities. When the count of unmapped LineStrings in a cluster reaches a significant number, it creates a representational line which is an average of all the paths in that cluster. This representational line formed is the new road that the framework proposes to a user. The correctness and accuracy of the newly formed road completely depends on the trajectory data. Third, the Decision Maker which identifies the new representational lines created and queries a volunteer about their validity. Based on the responses obtained from the users, the module decides to add this new path to the existing road network or it discards the proposed road.

3.1 Map Matching algorithm

We introduce a new topological map matching algorithm. The algorithm takes a GPS trajectory 'G' as input. A GPS Trajectory can be defined as a set of GPS coordinates obtained from a GPS device of a moving vehicle. The value |G| represents the number of points present in the trajectory. The path traveled by the GPS trajectory on the road network is known as the resultant path. The algorithm initially starts by finding the nearest roads on the respective map which are within the radius ε of the first point in the trajectory. The closest one will be considered as the resultant path. Then it checks if the next point in the trajectory is within the distance ε of the current resultant path. It will continue until next 'k' points i.e., point P_{i+k} ($i+k < |G|$) which does not match to the current road. It will again find out nearest paths for point P_{i+k} ($i+k < |G|$). The road among the nearest one which is adjacent, i.e., it sharing an edge with the previous resultant path, will be considered as a match.

In some cases, where the density of points in the trajectory is low, or when roads are connected by many smaller line-strings in between, we might not find roads which are adjacent to each other. In such cases, we can find if the roads are separated by one or two links. If this is true, we may include those paths or else we can consider P_{i+k} ($i+k < |G|$) and its previous point as outliers and those passed to the next step to verify if the same pattern is repeated in the coming trajectories.

If GPS data is noisy and of low sampling rate then this method may not return accurate resultant path. This is because, when the GPS data is farther than the said distance ε then the algorithm classifies them as outliers instead of matching them to the road network. This will affect the accuracy of the representational lines formed from the clusters in the next step.

3.2 Trace Analyzer

Trace Analyzer is responsible for clustering different outlier LineStrings returned by map matching module and forming a representative line. Similar unmapped LineStrings are compared with each other to check if any pattern or enough evidence is available using which we can infer a new road. To detect a new path, we need to find if a particular GPS segment, which is not part of the actual road network, appears frequently from many GPS trajectories. When the trace analyzer receives unmapped road as input, we build a cluster with a Minimum Bound Region (MBR) of width 20 meters on each side of the outlier LineString. The new outlier LineString is matched with existing clusters and, upon matching, is added to that cluster or else a new cluster is initialized. The presence of many outlier LineStrings in a cluster serves as evidence for a new road. So, to decide if a cluster should be considered to build a representational line, we need a parameter 'min cluster size'. If the number of unmapped trajectories in the cluster reaches 'min cluster size', then the representative line of that cluster is considered to be a potential new road which is marked as 'future roads' by the framework. If the newly formed road is within the distance of 10 meters on the existing road network, then those roads will be forwarded to the 'Decision Maker' module, to get user opinion on it. When the newly formed road is not within such range, the framework will wait for more evidence to completely join the road to the road network.

The value of 'min cluster size' should vary w.r.t the quality (i.e., noise) and frequency of the data submitted to the framework. A higher value of 'min cluster size' will delay the detection of new roads as it waits for more evidence before proposing a new road. Lower values of 'min cluster size' when the data is noisy results in faster updates with erroneous suggestions.

In this paper, we have implemented the trajectory clustering algorithm 'TRACLUS' [8]. All the parameters used in 'TRACLUS', are set to their optimal values as suggested in the original paper. We have compared its performance with an alternative approach based on image processing [4].

3.3 Decision Maker

The decision maker gets a representative line from the Trace Analyzer and queries a user to validate the new road. It will add it to the existing road network based on the response provided by the user. Decision can be taken based on user group's opinion, in a crowdsourcing setting, so that we can be more robust w.r.t. errors made by single users. The end-user can be the general crowd who have knowledge of that locality/city or, in ideal situations, the users who have uploaded the trace. A user may accept or reject the new road. If the users accept a new road, we can also ask them to provide additional information like road width, number of lanes, road name, and road type, etc. If the user rejects a proposed road, it is stored to avoid future errors.

4 EXPERIMENTS

We performed an experimental comparison of different modules discussed in this paper. To test the accuracy of the proposed map matching algorithm, it is compared with the HMM map matching algorithm discussed in [11]. To test the accuracy of the framework, existing roads from the maps are deleted and the GPS traces are fed as input to the framework. The similarity between the proposed roads and the previous, original roads is calculated.

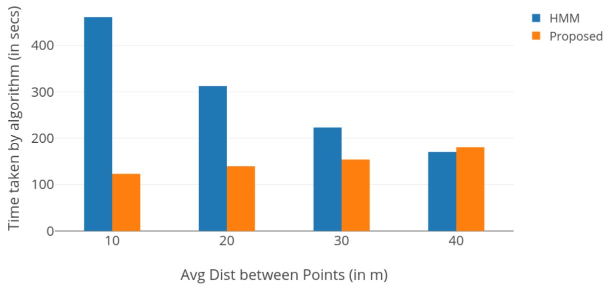

Figure 2: Comparison of Runtime for Seattle Dataset [11]

4.1 Comparison of Map Matching Algorithms

We have compared the proposed algorithm with the state-of-the-art Hidden Markov Model (HMM) map matching algorithm proposed in [11]. In the implementation of HMM map matching algorithm, the shortest distance between different points of a trajectory is required to calculate the transition probabilities. It is a resource and time intensive task. To speed up the process, shortest distance between nodes is pre-calculated and is used in the algorithm to calculate transition probability. Time to pre-calculate the shortest path is not included in the below statistics. The actual path taken by the GPS trace is known as 'Ground Truth' of the data and is calculated by visually analyzing the path. Results for both algorithms are compared with the 'Ground Truth'.

For the following experiments, we have considered 4000 points from the Seattle data set [11] and classified them into four categories of different minimum distances between the points in the dataset. In figure 2, we can observe that the runtime of the proposed algorithm is faster than HMM algorithm when the distance between the GPS points is small. This is because, when the average distance between points is small, there are many states and HMM needs to calculate their transition probability. Figure 3 and 4 show the comparison of precision and recall of both algorithms on Seattle Dataset respectively. Another quantitative metric used by [11] to compare map matching algorithms is Reported Error. Reported error is the ratio of length of road erroneously added and subtracted to the resultant path to length of the correct route. Figure 5 compares the reported error of both algorithms. Our proposal is shown to be much faster with very little loss in the quality metric.

The run time of the proposed algorithm is more than the HMM algorithm when the average distance between the points is 40 meters. This is because, when the points are distant from each other, the proposed algorithm has to verify if they are connected by any intermediate roads to ensure the continuity. This is an expensive operation and is computed on the fly. The preprocessing step for HMM algorithm took around 4 hours to completely calculate the shortest distances between the given GPS points.

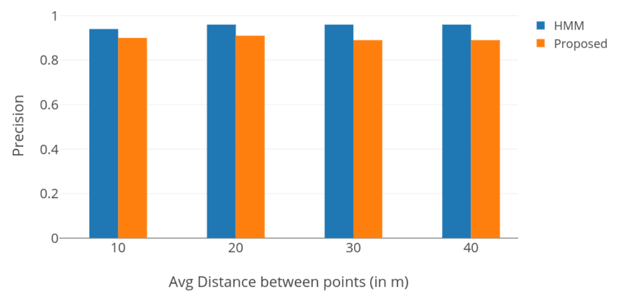

Figure 3: Comparison of Precision for Seattle Dataset [11]

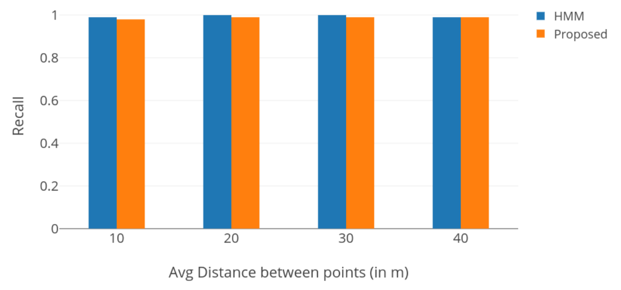

Figure 4: Comparison of Recall for Seattle Dataset [11]

4.2 Analysis of 'min cluster size' parameter

In Section 3.2, we have discussed that the 'min cluster size' of each cluster is the threshold which will decide if a representational line can be formed. In order to understand the effect of this parameter, we have run the entire framework for different values of 'min cluster size'.

To run this experiment, we have considered the map of Chicago and GPS trajectories[2] of 3 days and divided each day into shifts of 3 hours. For every shift, we have calculated the length of correct and incorrect roads proposed by the framework. Fig 6 and 7 show the length of the road correctly and incorrectly fetched respectively for different values of 'min cluster size'. The dotted line is the sum of the length of the roads that are deleted from the road network.

In Figure 6, we can observe that the lower values of 'min cluster size' were able to retrieve all the roads quickly but the length of incorrectly fetched roads is higher than for the later values. The lower value of 'min cluster size' implies that even with minimum evidence, we can consider the outliers as the potential new roads. For the noisy data, we might end up suggesting false positives as the new roads.

4.3 End-to-end Framework

In order to demonstrate the end-to-end quality of the framework, we have considered the map of Chicago and deleted two roads (in blue) from the existing road network, as shown in Figure 8. The framework is supplied with the GPS traces of 3 consecutive days

[2]https://www.cs.uic.edu/bin/view/Bits/Software

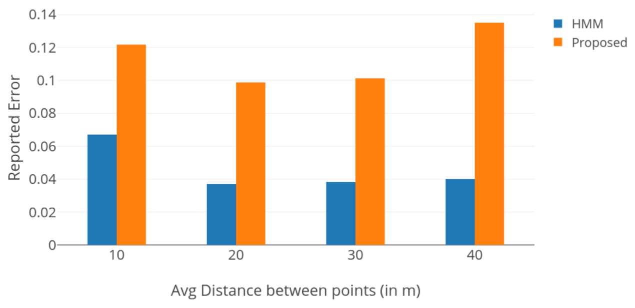

Figure 5: Reported error for both algorithms

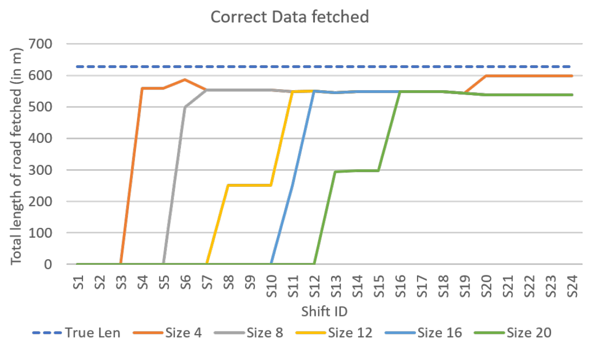

Figure 6: Length of Correct roads fetched

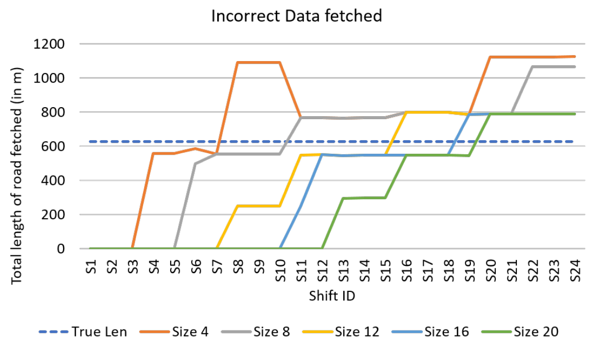

Figure 7: Length of Incorrect roads fetched

as input (Figure 9). At the end of the first step, the map matching algorithm returns the outliers from the traces which it has accepted as input. Then the outliers are clustered based on the similarity and distance between them. The value 'min cluster size' is set to 8 and We could successfully retrieve both the roads with high accuracy, as shown in Figure 11. The quality of the newly created roads can be assessed by visual verification or more precisely by using quantitative methods like Hausdorff distance and Frechet distance [3], as reported in Table 1. These figures are low and indicate that the proposed and deleted roads are very similar.

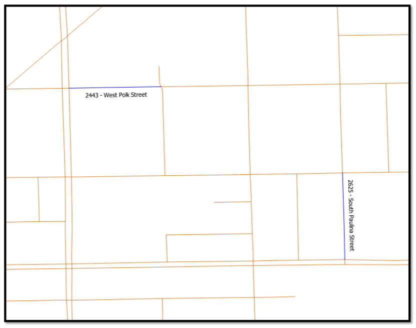

Figure 8: Roads (in blue) deleted from the road network

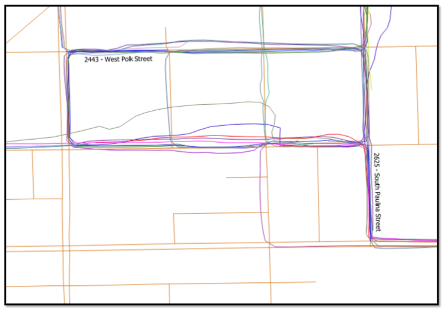

Figure 9: GPS traces for 3 days

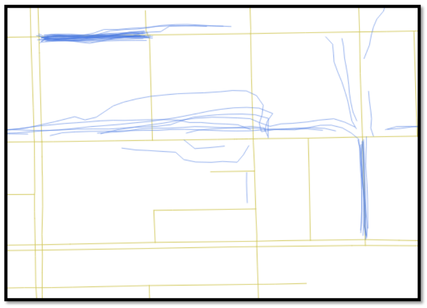

Figure 10: Outliers returned by the Map Matching Algorithm

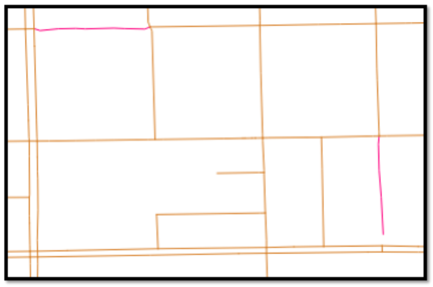

Figure 11: Suggested potential new roads (in Pink)

Road ID	Frechet Distance	Hausdorff Distance
2443	0.00245	0.00004
2625	0.00024	0.00024

Table 1: Similarity measurement of the missing roads

5 CONCLUSION

In this paper, we presented a framework that automatically detects new roads on a road network with high accuracy. The proposed framework is apt for the OSM use case, where a huge number of volunteers map the world and upload their GPS traces. The workload of updating maps can be delegated into an interactive process, which leads to minimum human intervention and provides an efficient way to maintain maps. In the future, we plan to extend the framework to detect lanes on a road. A preliminary solution can obtain road lane information from the end user who asserts the newly proposed road. Furthermore, we will extend the framework to detect roundabouts. In addition, we will extend the system to support data collected with a very low sampling rate. That can enhance the accuracy of the roads detected since this kind of data exists in practice.

REFERENCES

[1] 2017. (Jul 2017). http://wiki.openstreetmap.org/wiki/AddressHunter
[2] Ziawasch Abedjan, Xu Chu, Dong Deng, Raul Castro Fernandez, Ihab F. Ilyas, Mourad Ouzzani, Paolo Papotti, Michael Stonebraker, and Nan Tang. 2016. Detecting Data Errors: Where are we and what needs to be done? *PVLDB* 9, 12 (2016), 993–1004.
[3] Philippe C Besse, Brendan Guillouet, Jean-Michel Loubes, and François Royer. 2016. Review and Perspective for Distance-Based Clustering of Vehicle Trajectories. *IEEE Transactions on Intelligent Transportation Systems* 17, 11 (2016), 3306–3317.
[4] James Biagioni and Jakob Eriksson. 2012. Map inference in the face of noise and disparity. In *Proceedings of the 20th International Conference on Advances in GIS.* ACM, 79–88.
[5] Rene Bruntrup, Stefan Edelkamp, Shahid Jabbar, and Bjorn Scholz. 2005. Incremental map generation with GPS traces. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE.* IEEE, 574–579.
[6] Xu Chu, Ihab F. Ilyas, and Paolo Papotti. 2013. Holistic data cleaning: Putting violations into context. In *ICDE.*
[7] Stefan Edelkamp, Damian Sulewski, Francisco C Pereira, and Hugo Costa. 2008. Collaborative Map Generation–Survey and Architecture Proposal. *Urbanism on track: application of tracking technologies in urbanism* (2008), 161–182.
[8] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. 2007. Trajectory Clustering: A Partition-and-group Framework. In *Proceedings of the 2007 ACM SIGMOD (SIGMOD '07).* ACM, New York, NY, USA, 593–604.
[9] Biwei Liang, Tengjiao Wang, Shun Li, Wei Chen, Hongyan Li, and Kai Lei. 2016. Online Learning for Accurate Real-Time Map Matching. In *Proceedings, Part II, of the 20th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume 9652 (PAKDD 2016).* Springer-Verlag NY, Inc., NY, USA, 67–78.
[10] MapCraft. 2017. (Jul 2017). http://wiki.openstreetmap.org/wiki/MapCraft
[11] Paul Newson and John Krumm. 2009. Hidden Markov Map Matching Through Noise and Sparseness. 336–343.
[12] Francisco Câmara Pereira, Hugo Costa, and Nuno Martinho Pereira. 2009. An off-line map-matching algorithm for incomplete map databases. *European Transport Research Review* 1, 3 (01 Oct 2009), 107–124.
[13] Keep Right. 2017. (Jul 2017). http://wiki.openstreetmap.org/wiki/Keep_Right
[14] Map Roulette. 2017. (Jul 2017). http://wiki.openstreetmap.org/wiki/MapRoulette
[15] Renchu Song, Wei Lu, Weiwei Sun, Yan Huang, and Chunan Chen. 2012. Quick Map Matching Using Multi-core CPUs. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL '12).* ACM, New York, NY, USA, 605–608.
[16] OpenStreetMap Wiki. 2014. Main Page — OpenStreetMap Wiki. (2014). http://wiki.openstreetmap.org/w/index.php?title=Main_Page&oldid=1060762