Vehicular Dead Reckoning Based on Machine Learning and Map Matching

Lucas de C. Gomes, *Student Member, IEEE,* and Luís Henrique M. K. Costa, *Senior Member, IEEE*
Universidade Federal do Rio de Janeiro - GTA/COPPE/UFRJ - Rio de Janeiro, Brazil
{gomes, luish}@gta.ufrj.br

***Abstract*—Global Navigation Satellite Systems (GNSS) are used today in various contexts as a source of data for several applications. They provide real-time positioning based on the transmission of electromagnetic waves from a satellite to a receiver, being subject to several factors. Some scenarios, such as canyons (urban or geographic), forests and tunnels, are challenging, since the coverage in them is unavailable or unreliable, producing rogue positioning information or no information at all. Thus, applications that demand high availability usually employ other sensors. Nevertheless, reducing the amount of such devices results in lower costs and energy consumption. Aiming to improve the reliability and availability of GNSS-based systems retaining cost-effectiveness, this work proposes a dead reckoning system, using the last known location and sensor data to infer the current position. The sensors employed here are largely available in commercial vehicles. We calculate the estimates using machine learning models and improving the results through a map matching procedure. The results, based on simulations with real GNSS and sensor data, indicate that the system is able to closely reproduce trajectories for over a minute. The obtained mean error is of approximately 19 meters, suitable for obtaining approximate locations in scenarios with unreliable satellite coverage.**

***Index Terms*—Dead reckoning, GNSS, inertial sensors, machine learning, map matching.**

I. Introduction

Today, geographic positions are an important information for various applications of our daily lives, such as locating people, objects or vehicles worldwide, step-by-step navigation systems (e.g. Waze) and traffic safety applications, such as collision prevention systems or traffic control systems [1], [2]. Probably the most common approach to get location information is using a Global Navigation Satellite System (GNSS). Nowadays, four of them are in operation: American GPS (Global Positioning System), Russian GLONASS (*Globalnaya Navigatsionnaya Sputnikovaya Sistema*, Russian for Global Satellite Navigation System), European GALILEO and Chinese BeiDou. These calculate the position of a receiver in Earth through the propagation time of electromagnetic waves, sent by one or more satellites to the receiver. Thus, several environmental and equipment-related factors affect their performance, such as the atmospheric layers, weather conditions, reflections on obstacles and inaccuracies on equipment hardware and software.

GNSSs are also subject to coverage losses in certain environments, such as forests, tunnels, and canyons (urban or geographic). In these conditions, communications between satellites and receivers are error-prone, harming or fully stopping the operation of applications that require this information, depending on the required availability and the duration of the downtime. For traffic safety applications and autonomous vehicles, this increases the risk of accidents. These factors motivate the adoption of additional sensing equipment in some applications [3]. Nevertheless, reducing the amount of sensors is convenient: it lowers installation and maintenance costs, the configuration complexity and the energy consumption.

Seeking to improve the availability of position data with a low-cost solution, this work proposes a Dead Reckoning (DR) system which provides real-time location for land vehicles during lacks of coverage, keeping the accuracy at adequate levels. The proposed algorithm estimates the current location based on the last known position and sensor data. The employed sensors are already installed in some car models. To relate the sensor readings to vehicle displacements, Machine Learning (ML) models for regression and clustering are trained with data extracted from a real scenario and used to provide estimations. These estimations are adjusted with a map matching algorithm, in order to improve their accuracy.

This paper is organized as follows. Section II discusses related work. Section III presents the used equipment, while Section IV presents our DR algorithm. Section V describes the data collection procedure and highlights the data set relevant features. Section VI discusses the obtained results. Finally, Section VII provides final remarks and discusses future work.

II. Related Work

Different works propose vehicular position estimation methods under faulty satellite communication.

The step-by-step navigator Waze has integrated the project Waze Beacons [4], which provides real-time positions and the possibility of sending alerts inside tunnels. Low-cost microcontrollers communicate with users' smartphones through Bluetooth: positions are estimated through metrics obtained in the procedure. This project is present in some cities, such as Haifa (Israel), Chicago (USA) and Rio de Janeiro (Brazil). Its disadvantage is the requirement of the installation and the upkeep of a fixed infrastructure.

Belhajem *et al.* [5] employ an Extended Kalman Filter (EKF) to predict positions with data from an accelerometer and a gyroscope. Using only the EKF, errors quickly accumulate. To compensate, a neural network or a support vector machine, adjusted by optimization techniques, computes correction factors. These models are trained and tested with real data, collected with a moving car. Even with corrections, errors still accumulate quickly, albeit with a lower intensity. The error surpasses 100 meters in under a minute. The average error varies between 23 m and 315 m in the north component and between 20 m and 304 m in the east component, depending on the circuit section.

Pinto Neto *et al.* [6] employ linear regressions as position estimators. Using wheel rotation frequency sensors, the type of trajectory is determined and the position is calculated by the model that was trained with data corresponding to that type of trajectory. As in [5], the issue is error accumulation, due to the reuse of past positions and estimates. To avoid it, a correction factor is dynamically updated and applied. Their system is tested with real GPS and sensor readings using a vehicle and a robot. For the vehicle, the mean error is kept below 5 m for a maximum of 80 seconds. However, after this duration, the error assumes a rapidly growing behavior.

Ahmed *et al.* [7] employ yaw rate, speed sensors and an inertial measurement unit. A particle filter determines the displacement and matches the estimates with a map that provides lane-level data. A DR procedure, using the speed and the yaw rate to calculate the displacement, is used when a lane change is detected by a high enough yaw rate value. Only the longitudinal error was evaluated. For this metric, at the initial 100 m (corresponding to 11.2 s at a mean speed of 32 km/h), the values are over 10 m, with a maximum of approximately 90 m. After this stretch, the error is lower than 1 m. This behavior is due to the time the algorithm's particles take to converge to more accurate values.

Our proposal shares similarities with these proposals, since it incorporates ML techniques and a map matching algorithm and, similarly to [6], it uses a family of regression models, each one being adjusted for each kind of trajectory. Nevertheless, the selection of the threshold sensor readings for choosing one of the models is automatic, operated by a clustering algorithm.

III. Equipment Setup

We have used GPS for positioning, although the approach can be adapted to any GNSS. Our system focuses on land vehicles, being designed for an OBU (On-Board Unit). In this paper, we use an off-the-shelf device, Cohda Wireless MK5 OBU, equipped with wireless communication interfaces, a GPS receiver and a connection with the CAN (Controller Area Network) bus, a cable network through which compatible equipment can access sensors and actuators installed in a car [8]. The vehicle provides, through CAN, its wheel rotation frequencies and its speed. As will be shown in the next sections, in a turn, the left and right wheels' frequencies differ, allowing the detection of curves. The speed sensor allows to determine the movement length, regardless of direction. Employing only sensors that are available in the vehicle is cost-effective and independent of external communications, mitigating scenarios with lacks of coverage. The employed sensors are available in vehicles equipped with Anti-lock Braking Systems (ABS) [9]. Table I lists the equipment characteristics. The sensors update rate is the average value; its uncertainty is the standard deviation, obtained by the analysis of the collected readings, described in the Section V.

Table I: Description of the used equipment.

Type	Name	Important Features
On-Board Unit	Cohda Wireless MK5	OBD-2 connector
GPS Receiver	U-blox M8N	Accuracy: 2.5 m
		4 Hz update rate
Vehicle	Peugeot 408 (2016)	CAN update rate:
		(49.9 +/- 0.2) ms

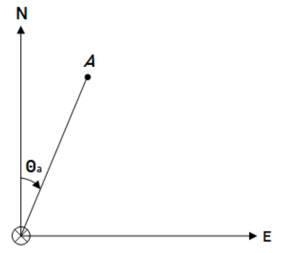


Figure 1: The NED (North-East-Down) model.

IV. Methodology

The coordinate system used with GNSS receivers, encompassing latitude and longitude, does not allow to separately treat linear and angular displacements. Therefore, we convert it to the NED (North-East-Down) coordinate system. It measures positions on a tangent plane in relation to a reference point, approximating the Earth as flat in a small area, which does not induce large errors for short distances [10]. Positions are represented on the horizontal and vertical axes in relation to the reference, as Figure 1 shows.

The movement of an object from the origin to point A has a *heading angle* of θ_a, components in the east and north directions, a speed and a duration. The relations between those elements are:

$$\Delta E = V sin\theta \; t, \tag{1}$$

$$\Delta N = V cos\theta \; t, \tag{2}$$

where ΔE, ΔN, θ and t are, respectively, the differences in the east and north directions, the direction and the duration. The speed is obtainable by a sensor and t is the time between two GNSS updates, i.e., the inverse of the receiver update frequency. Our goal is to calculate the displacements in each direction, in order to adjust the last known position. For that, the heading angle is necessary, requiring two positions. The scenarios of concern are GNSS-denied environments; directly obtaining it, thus, is unfeasible. However, we may estimate its variation through the wheel rotation frequencies:

$$\Delta\theta = f(f_{rl}, f_{rr}), \tag{3}$$

where f_{rl} and f_{rr} are the rotation frequencies of the rear left and rear right wheels. Using a Linear Regression (LR), we approximate it by:

$$\Delta\theta \approx af_{rl} + bf_{rr} + c, \tag{4}$$

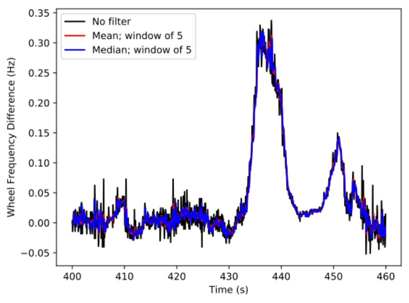


Figure 2: Filtering procedures in a trajectory with a turn.

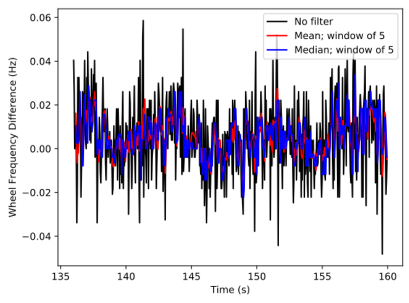


Figure 3: Filtering procedures in a straight line trajectory.

where a, b, and c are determined through the LR via the least squares method [11]. The new heading angle is obtained by adding $\Delta\theta$ to the previous value. As will be seen later, our data has different amounts of readings for straight lines and turns. To prevent this imbalance from harming the prediction of turns, a set of linear regression models was created, one for each type of reading. Readings are grouped based on the absolute values of their wheel frequencies, representing different movement types (such as straight lines, light turns and intense turns); each group is treated by the linear regression model associated with it. Therefore, each model is fine-tuned for a type of movement. For dividing the readings, the K-means clustering algorithm was employed, which finds a set of clusters with an arbitrary cardinality [11]. We implemented the initialization criterion k-means++ [12], which finds the set of centroids that are the most distanced from each other based on the given data. The exact number of movement types is investigated in Section VI.

To improve the estimates and limit error accumulation, we use a map matching (MM) algorithm afterwards. It matches given geographic positions to a map, yielding corrected ones on defined stretches, such as streets or highways, close enough to the given location. Here, we use the open source implementation of OSRM (Open Source Routing Machine) [13], with OpenStreetMap [14]. For offline availability, the map of the region of the field tests and OSRM are locally installed. In order to avoid inaccurate matches, if there is more than one possible matching, our estimate is not modified.

Sensor readings are pre-processed by a filter. Since they are more abundant than positions, as reported in the next section, we may associate multiple previously received sensor readings to one position in a cumulative procedure. Thus, using aggregate calculations (e.g. mean and median), the noise is reduced. Figures 2 and 3 give examples of filtering in a turn and in a completely straight stretch. Both filters employ a sliding window of 5 readings. The median filter produces smoother variations, being, thus, selected for wheel rotation frequencies and speeds.

V. Data Collection and Analysis

We collected, from a real scenario - the campus of Federal University of Rio de Janeiro -, positions from a GPS receiver and wheel frequency and speed values from the CAN bus. For that, the OBU was connected to the vehicle's OBD-2 connector and periodically fetched those values while the vehicle traversed the area. During the procedure, the conditions were ideal: it was a sunny day of cloudless sky and there was no obstacles which could induce coverage losses, as the region is a predominantly open area. Given that atmospheric conditions and obstacles affect the GPS accuracy, these conditions are the most adequate to collect data for training our models. Table II briefly describes the collected data, whereas Figure 4 exhibits the path using Google Maps. The trajectory has approximately 3.6 km and was traversed counterclockwise, with speeds up to 51 km/h.

Table II: Main characteristics of the collected data set.

GPS Reading Count	2107
CAN Reading Count	13614
Total Size	796.8 kB
Features	Timestamp (GPS), Timestamp (CAN), Latitude, Longitude, Left and Right Rear Wheels Frequencies, Speed

Demonstrating the capability of inferring directions and intensities of turns, Figure 5 indicates the wheel frequency differences on the target. Left turns can be identified by negative values (shades of blue), right turns are deduced by positive values (shades of red) and straight movements are indicated by low absolute values (brown). Figure 6 shows heading variation values. Values lower than -1.5° are displayed in blue, values greater than $+1.5^{\circ}$ are represented by red dots and values between those limits are exhibited in black. This criterion was employed solely to improve the visualization. Again, left turns, right turns and straight trajectories are represented by negative values, positive values and low or null heading values, respectively. Some colored dots can be seen in apparently straight stretches: this may indicate lane changes or GNSS position errors. A visual comparison of both maps signals the correlation between the heading variations and wheel frequency differences, which has allowed conceiving the framework discussed in the previous section.

VI. Performance Results

We apply our algorithm to the data described previously. To verify its robustness to data sets with different features (such as more turns or longer straight lines), cross-validation, a procedure commonly used to evaluate ML algorithms, is

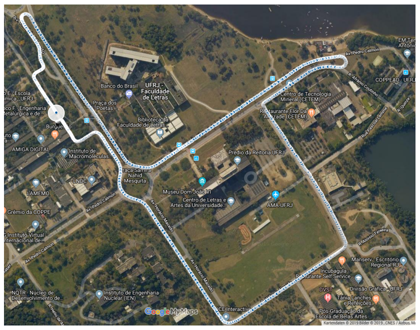


Figure 4: Trajectory for the data collection procedure (Source: Google Maps).

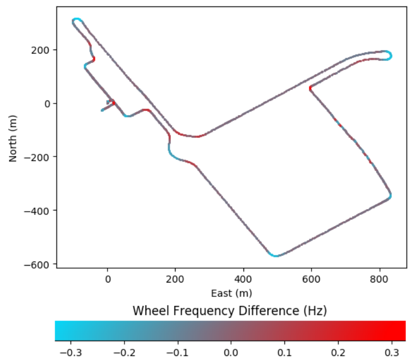


Figure 5: Heat map indicating the wheel frequency differences.

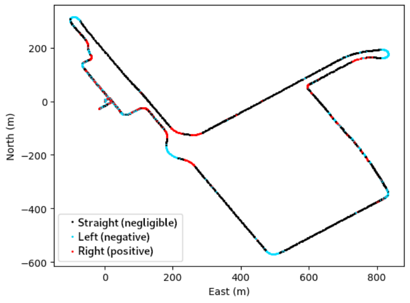


Figure 6: Map highlighting the heading variations.

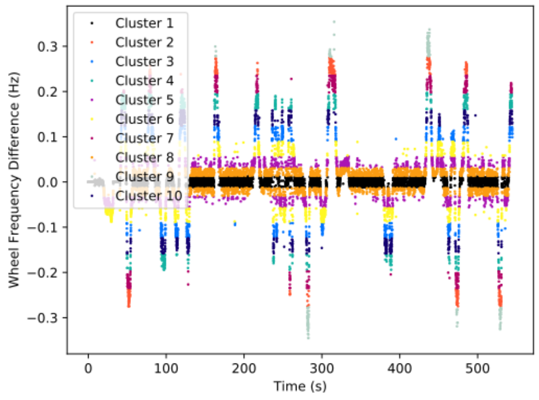


Figure 7: Final clusters after running K-means for 10 clusters.

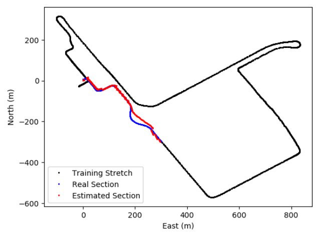
used. The data was split into four sections of consecutive readings, each having the same size. Three are chosen as training data - used to adjust the models to the data set, by calculating a relation between inputs and outputs -; the remaining one is used as test data: its outputs (geographical positions) are estimated from its inputs and compared to their real values. Such division enables four different permutations. The simulated scenario, for each permutation, is: given the last position and heading angle before the test section, estimate the trajectory only with the sensors.

Several configurations in each permutation were analyzed, by varying the amount of clusters that K-means should find (n_c) and the period of map matches, defined by the amount of estimates before a new correction (n_m). n_c and n_m were varied from 1 to 10. The best case is the one with the lowest mean error: $n_c = 10$ and $n_m = 10$, with a global mean error of 19.1 m and mean errors of 8.3 m, 8.3 m, 28.9 m, 30.9 m in Permutations 1, 2, 3 and 4, respectively.

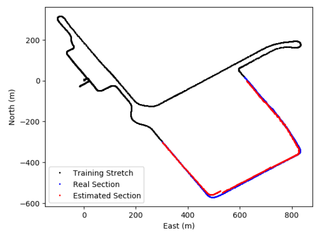
Figure 7 shows the clustering results over the wheel frequency differences of the data set for this case. Figures 8a to 8d exhibit, for each permutation, the training and test sections and the predicted trajectory for the test section, for the best configuration. The training section, the real positions of the test section and the predicted values are, respectively, in black, blue and red. Figure 9 shows the error behaviors in each permutation, for the best configuration. The test section of Permutation 1 presents the lowest vehicle speeds, as it corresponds to the start of the experiment: as a side effect, when the vehicle is quasi-stationary, the system suffers more with GNSS imprecision. The error behavior indicates it grows over time, although with a low intensity, and is dependent on the sharpness of turns - more intense turns cause greater errors - possibly because of the lower amount of readings in turns. Even without imbalances in the data sets used to train each linear regression, due to the clustering, more readings lead to a better model quality. This accumulation is expected and seen in related work: dead reckoning systems employ recent positions or estimates; when using estimates, their error is also fed to the system, leading to its growth over time. The error component that is perpendicular to the street axis is contained by the map matching, when it finds the correct matching street. However, GNSS position shifts have induced inaccurate matches in Permutations 1, 3 and 4, inducing controlled error increases. Later, in those cases, correct matches are done, allowing the system to recover the accuracy. The error is kept below 70 meters after a 140-second lack of coverage in Permutation 4, a scenario with sharper turns and lower speeds at its end, and reaches lower maximum values in other scenarios.

VII. Conclusion

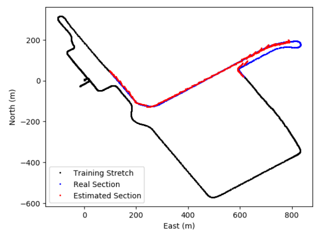
This paper proposed a cost-effective and self-contained DR system for vehicles, which only employs sensors already available on ABS-equipped vehicles. It uses linear regression models fine-tuned for different movement patterns and corrections



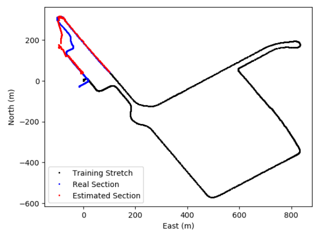
(a) Permutation 1.



(b) Permutation 2.

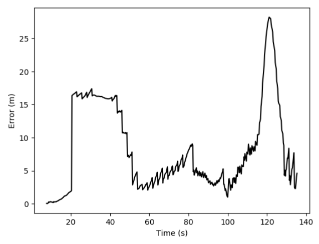


(c) Permutation 3.

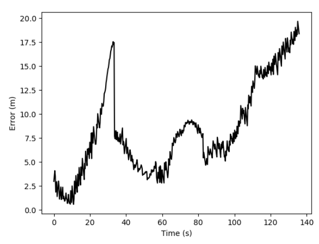


(d) Permutation 4.

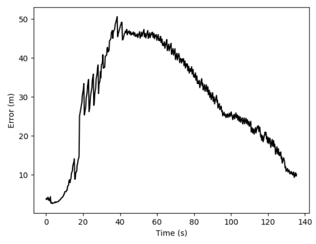
Figure 8: Training stretches and comparisons between the real and estimated test sections for each permutation.



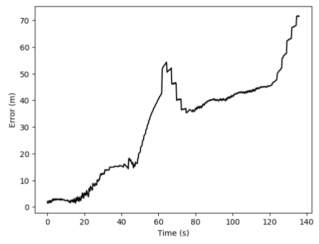
(a) Permutation 1.



(b) Permutation 2.



(c) Permutation 3.



(d) Permutation 4.

Figure 9: Position errors in meters for each permutation.

from a map matching algorithm. The estimations hold fidelity with the real paths taken in every setting. One may observe that the error accumulation is mostly manageable over a long period at speeds commonly employed by drivers. Due to this behavior and the achieved mean error, this system is suitable for obtaining approximate locations, tracking vehicles, people or objects and navigation. The accuracy shows improvements over the levels on [5]. Our system does not need an "initial calibration", during which the error is high until an adequate improvement on the estimation quality is done, which occurs in particle filter implementations such as [7], showing our system is more suitable in scenarios that demand quicker responses.

Future works will focus on improving the accuracy: we intend to test other filters for the sensors and adjust the map matching procedure to include lane-level information, since our approach adjusts the positions to the streets' centers in the axis which is perpendicular to the street; this might have contributed to the errors and the inaccurate matches seen here.

Acknowledgment

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, CNPq, FAPERJ, and FAPESP Grants 15/24494-8 and 15/24490-2.

References

[1] J. B. Pinto Neto, L. C. Gomes, E. M. Castanho, M. E. M. Campista, L. H. M. K. Costa, and P. C. M. Ribeiro, "An error correction algorithm for forward collision warning applications," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Nov 2016, pp. 1926–1931.

[2] M. Ferreira, R. Fernandes, H. Conceição, W. Viriyasitavat, and O. K. Tonguz, "Self-organized traffic control," in *Proceedings of the Seventh ACM International Workshop on VehiculAr InterNETworking*, ser. VANET '10. New York, NY, USA: ACM, 2010, pp. 85–90.

[3] M.-F. Tsai, Y.-C. Chao, L.-W. Chen, N. Chilamkurti, and S. Rho, "Cooperative emergency braking warning system in vehicular networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2015, no. 1, pp. 1–14, 2015.

[4] "Waze Beacons." [Online]. Available: https://www.waze.com/en/beacons

[5] I. Belhajem, Y. B. Maissa], and A. Tamtaoui, "Improving low cost sensor based vehicle positioning with machine learning," *Control Engineering Practice*, vol. 74, pp. 168 – 176, 2018.

[6] J. B. P. Neto, N. Mitton, M. E. M. Campista, and L. H. M. K. Costa, "Dead reckoning using time series regression models," in *Proceedings of the 4th ACM MobiHoc Workshop on Experiences with the Design and Implementation of Smart Objects*, ser. SMARTOBJECTS '18. New York, NY, USA: ACM, 2018, pp. 6:1–6:6.

[7] H. Ahmed, M. Tahir, and K. Ali, "Terrain based gps independent lane-level vehicle localization using particle filter and dead reckoning," in *2016 IEEE 84th Vehicular Technology Conference (VTC-Fall)*, Sep. 2016, pp. 1–5.

[8] "Controller Area Network (CAN) Implementation Guide – Analog Systems," 2017. [Online]. Available: http://www.analog.com

[9] A. Daiss and U. Kiencke, "Estimation of vehicle speed fuzzy-estimation in comparison with kalman-filtering," in *Proceedings of International Conference on Control Applications*, Sep. 1995, pp. 281–284.

[10] M. Grewal, L. Weill, and A. Andrews, *Global Positioning Systems, Inertial Navigation and Integration*. John Wiley & Sons, Inc., 2001.

[11] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning - Data Mining, Inference and Prediction*. Springer, 2009.

[12] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics*, 2007.

[13] D. Luxen and C. Vetter, "Real-time routing with openstreetmap data," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. GIS '11. New York, NY, USA: ACM, 2011, pp. 513–516.

[14] "OpenStreetMap." [Online]. Available: https://www.openstreetmap.org