Determining the Location of Objects Using a Vision-System Sensor and CPM

Silas Lobo, Evelio Fernández, Christian Facchi

Abstract— **Vehicular Ad Hoc Networks (VANETs) are expected to be the next big step towards safer road transport, supporting applications to exchange information between vehicles according to ETSI ITS G5. One of the major benefits is to share data related to the onboard sensors, affording to aware surroundings vehicles about objects that are not in their line-of-sight. Using this data, all network participants can develop a Dynamic Map with all relevant obstacles. This paper presents an approach, named GDO, to assist a map creation applying the MobilEye as onboard sensor. To this end, several tests were implemented and evaluated.**

Keywords— **VANET, Collective Perception Message, CPM, Perceived Object, Car2X, MobilEye.**

I. Introduction

Vehicular Ad-hoc Networks (VANETs), also known as Car2X, is a cooperative wireless network, which establishes a communication environment for road participants. This communication may be between vehicles (V2V) and between vehicles and all traffic participants (V2X) [1]. Features provided by VANETs are concentrated in the non-safety and the safety [2] branches. According to the safety area, VANETs are structured to provide relevant information, identifying events that might cause hazard situations [3]–[6].

Current vehicles are equipped with on-board sensors, as LIDARs, radars and cameras which measure at real-time the vehicle's surrounding environment. Consequently, VANETs can assist nearby vehicles to exchange sensor's information with each other, minimizing the limitations of the sensors. This communication may provide a safer traffic, creating accurate knowledge about the surroundings obstacles. Furthermore, it extends the environment's perception for all road users, creating a collective perception.

European Telecommunications Standards Institute (ETSI) has published the ETSI TR 103 562 [7] in order to standardize the collective perception of messages within Car2X communication. This technical report is in the last step to become a standard and introduces the Collective Perception Message (CPM), a message generated by vehicles, gathering data from all sensors and sharing information with nearby stations. This allows vehicles to enrich their own environment perception based on other vehicles perception [8]. The increased perception assists vehicles to create a complete Dynamic Map (DM) including not only objects detected by its own sensors, but also inserting obstacles detected by other vehicles. DMs improves the reliability of the environment perception and increases the awareness by removing potential blind spots.

Thus, this work proposal is to develop a tool to assist the development of a DM, computing the objects' location using the MobilEye[1] camera as an onboard sensor, and then evaluate the distance errors between measured data and a database. This tool extracts the camera's parameters to attend CPM according to ETSI TR 103 562, and it is extended to calculate the object's location.

The remainder of this paper is organized as follows: CPM and ETSI TR 103 562 are introduced in Section II. After presenting the research environment in Section III, the development of this research is described in Section IV. Section V presents results and Section VI concludes the paper and discuss future work.

Silas Lobo, Department of Electrical Engineering, Universidade Federal do Paraná (UFPR), Curitiba-PR and Faculty of Electrical Engineering and Information Technology, Technische Hochschule Ingolstadt (THI), Ingolstadt, Germany, e-mail: SilasCorreia.Lobo@thi.de; Evelio Fernández, Department of Electrical Engineering, Universidade Federal do Paraná (UFPR), Curitiba-PR, e-mail: evelio@eletrica.ufpr.br; Christian Facchi, Faculty of Electrical Engineering and Information Technology, Technische Hochschule Ingolstadt (THI), Ingolstadt, Germany, e-mail: Christian.Facchi@thi.de

II. Collective Perception Message (CPM)

Previous studies have shown the benefits of collective perception in the vehicular area [9]–[11]. This approach is enhanced by Car2X applications partly standardized by ETSI TR 103 562 [7]. ETSI brings the model where cars interact with each other through VANETs, informing nearby vehicles of detected objects, as cyclists, pedestrians, and other vehicles by their on-board sensors [10]. As part of the standard, CPM has been introduced to exchange the sensors' information. The message architecture consists of Intelligent Transport Systems (ITS) Protocol Data Unit Header (ITS PDU Header) and CPM parameters [7].

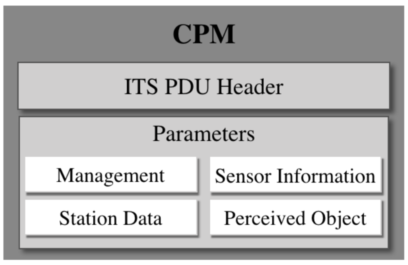

Fig. 1. CPM Structure [7]

Figure 1 depicts the basic structure of a CPM presented by ETSI. *Management Container* and *Station Data Container* provide information related to type of the sender station, if it is a vehicle or a Road-Side-Unit (RSU), actual position,

[1]https://www.mobileye.com

orientation angles, vehicle's dynamic, and heading. *Sensor Information Container* inserts information regarded to sensors attached to the sender, e.g. LIDAR, radar, camera and their characteristics, such as ranging, opening angle, and location where it is placed in the ITS-S. *Perceived Object Container* brings a list of detected objects, detailing their dimensions, relative position, speed, acceleration and other data described in Table I.

III. Determining the Location of Detected Objects

The current location of an object detected by an on-board sensor is defined based on the sensor's environment. To expand this view to other vehicles, a common base location has to be implemented. This study considers latitude and longitude based on the World Geodetic System 1984 (WGS84) to represent the locations [12].

Geolocation of Detected Objects (GDO) is a system composed by two algorithms and is introduced by the authors. The proposed system collects data from an on-board sensor, fills parameters related to the CPM, intending to computes geolocation of detected objects. Furthermore, a GNSS Receiver and an inertial measurement unit (IMU) are also integrated to the GDO. Figure 2 shows the system dependency implemented in this research work.

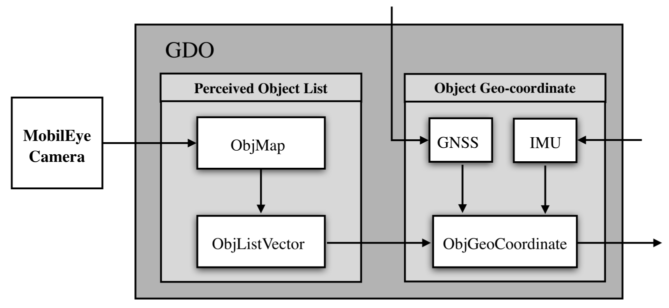

Fig. 2. Geolocation of Detected Objects Structure

The implemented on-board sensor is a MobilEye Camera 660, the GNSS Receiver is an Ublox M8 GNSS Receiver [13], and IMU is the Variense VMU931 [14]. All these devices have been integrated in the GDO to perform the tests.

The MobilEye camera is a vision-system sensor connected to the vehicle's Controller Area Network (CAN) [15], giving the information listed in Table I for each detected object. This table also gives a comparison between the information covered in ETSI TR 102 562 for *Perceived Object Container* and the object's characteristics provided by the camera, e.g. object dimensions.

The Ublox M8 GNSS is a high-precision unit equipment combined by an antenna located in the rooftop of the test vehicle, and a receiver connected to car PC calculating the latitude, longitude, vehicle's heading, and direction of motion. The vehicle's heading describes the angle between the North and the direction where the front of the car is pointed. Moreover, the heading of motion corresponds to the angle between the North and the vehicle's direction of movement and could represent forward and backward directions. The IMU VMU931 provides the vehicle's attitude as pitch and roll angles.

TABLE I
Object's Elements comparison between ETSI and MobilEye for CPM.

Object Characteristic	**ETSI**	**MobilEye**
`objectID`[2]	x	x
`sensorIDList`	x	x
`timeOfMeasurement`[2]	x	
`objectAge`	x	x
`objectConfidence`[2]	x	x
`xDistance`[2]	x	x
`yDistance`[2]	x	x
`zDistance`	x	
`xSpeed`[2]	x	x
`ySpeed`[2]	x	x
`zSpeed`	x	
`xAcceleration`	x	x
`yAcceleration`	x	
`zAcceleration`	x	
`yawAngle`	x	x
`planarObjectDimension1`	x	x
`planarObjectDimension2`	x	x
`verticalObjectDimension`	x	
`objectRefPoint`[2]	x	x
`dynamicStatus`	x	x
`classification`	x	x
`matchedPosition`	x	x
`timeOfDetection`		x

The car PC runs Ubuntu[3] 18.04, which connects all the devices previously described. The IMU and GNSS were connected via USB. MobilEye is connected to the vehicle's CAN, and a bridge to connect the CAN BUS of the car was implemented and attached to the computer. Furthermore, car PC runs the Robot Operating System (ROS) Melodic Morenia[4] creating a development environment to gather the sensor station and the car.

IV. GDO Development

Intending to calculate the geoposition of the detected objects, two algorithms have been developed. The first algorithm creates an object list for the *Perceived Object Container* based on MobilEye camera. The second algorithm gathers data from IMU, GNSS, and the object list to compute latitude and longitude for each object.

A. Perceived Object List

A ROS node was developed within the car to create a *Perceived Object List* according to [7]. This algorithm is divided into two modules, a module to store all data from the MobilEye camera, and a node to evaluate which data is relevant to be sent further.

The `objMap` is the container module, where every obstacle detected by the camera is stored. The *key value* is associated

[2]Mandatory element in CPM
[3]https://ubuntu.com
[4]https: //www.ros.org

with the `objectID` provided by the camera to keep only the last measured value from an object. Moreover, the *mapped value* stores all data related to this object.

The `objectListVector` is an evaluation module, which iterates over the `objMap`, and is triggered in a 10 Hz rate. For each callback, it compares the object's `timeOfDetection` with the actual running time. Objects detected in less than 200 milliseconds are pushed into the `objectListVector`, otherwise, they are removed from the `objMap`. Furthermore, `objectListVector` is published, and `objMap` is erased. Algorithm 1 describes the steps implemented to develop the *Perceived Object List*.

Algorithm 1: Perceived Object List Algorithm

```
Result: Create Perceived Object List
1  initialize $timeToSend$;
2  initialize ObjMap;
3  while $timeToSend = false$ do
4      wait for object detection from the camera;
5      if it is a new object then
6          add this object to objMap with key = objID;
7      else
8          overwrite previous information with new data;
9      end
10     update ObjMap;
11 end
12 if $timeToSend = true$ then
13     initialize ObjListVector;
14     if $(timeOfDetection - actualTime) > 200ms$
         then
15         delete object from ObjMap;
16     else
17         insert object to ObjListVector;
18     end
19     iterate over the ObjMap until the end;
20     publish ObjListVector;
21     erase ObjMap;
22 end
```

As object inclusion in the *Perceived Object Container* stills an open question in [7], this study has implemented the time of 200 milliseconds to evaluate objects, considering that objects detected beyond this time have a lower impact on the environment around the vehicle.

B. Object Geoposition

Object Geoposition is an algorithm developed as a ROS node to compute the latitude and longitude for each object published by `objListVector`. This algorithm consists of three modules, two callbacks to maintain the last value of GNSS and IMU data, and one to calculate the geo-coordinate values.

The *GNSS* module is a callback function to update values of latitude, longitude, and vehicle's heading. These values are updated at a frequency of 10 Hz. *IMU* module is also a callback function assigned to the IMU VMU931 output and receives the attitude data, which is the pitch, roll and yaw angles. As these values are provided in quaternion format, this function also implements a conversion to Euler angles [16] related to the pitch angle and stores the value for further usage.

GeoCoordinate is the main module of this node implementing an offset correction, and computing objects latitude and longitude. *RefPointOffset* is a sub-module to calculate the geoposition of the reference point. This offset is required, due to, GNSS' latitude and longitude are assigned to the antenna location. The antenna is located on the back part of the rooftop of the vehicle, and the reference point of the MobilEye is in the middle of the front bumper. The offset developed in this step is expressed in Figure 3, comparing the position of the GNSS antenna on the rooftop of the vehicle reference point in the middle of the front bumper.

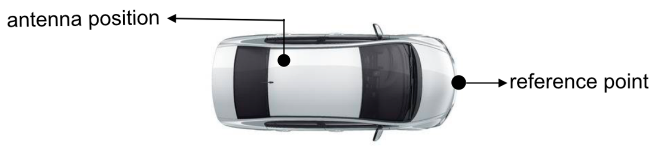

Fig. 3. Reference Position Offset

Whenever a vehicle is driving up or down a hill, the distance between the vehicle and an object is reported by the MobilEye with the euclidean distance. Yet, an offset to compute the geodesic distance has to be deployed. This correction is based on the pitch angle and levels of vehicle and object. The sub-module that implements this functionality is *RampImpact*.

The MobilEye camera provides the angle between the vehicle's reference point and the detected object. When the obstacle is detected to the right of this point, MobilEye represents a positive angle. However, if the obstacle is detected to the left of the reference point, it reports a negative angle. Thereby, the ϕ angle is the addition of the vehicle's azimuth and the angle reported by the camera. Figure 4 reports this situation, where α is the azimuth angle of the vehicle and β is the angle between the vehicle and the obstacle. In this case example, $\phi = \alpha + \beta$.

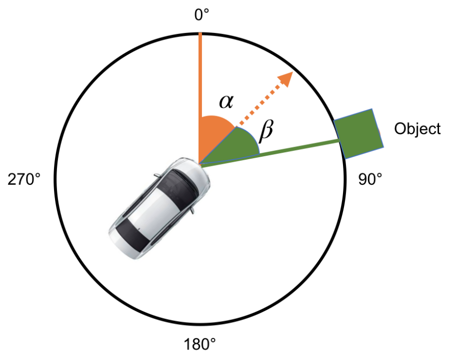

Fig. 4. Angle representation between the vehicle's direction of movement, azimuth, and detected object

Thus, to define object's latitude and longitude the Vincenty's Formulae [17] theorem was implemented. This theorem demands vehicle's reference point, latitude and longitude, the distance between this point and the object, Earth's radius, object's azimuth angle, and the angle between ref. point and object. Algorithm 2 describes the method applied to calculate the object's geoposition, publishing latitude and longitude for each object presented in `objListVector`.

Algorithm 2: Object Geoposition Algorithm

```
   Result: Compute Object Geo-Coordinate
 1 keep IMU data updated;
 2 keep GNSS data updated;
 3 if received a new objListVector then
 4     calculate refPointOffset;
 5     calculate Euclidean distance;
 6     calculate ramp impact;
 7     calculate $\phi$ angle;
 8     apply Vicenty's formulae;
 9     publish Object's Latitude and Longitude;
10 end
```

V. GDO Evaluation

GDO was implemented in the car PC and tests have been run. Firstly, a test scenario was designed, consisting of several objects such as pedestrian and other vehicles crossing in front of the test vehicle at distances from 0.84 to 5.75 meters. In test, the distance between the vehicle's ref. point and the object was measured for each object by means of the Haversine Formula [18], which takes in consideration the latitude and longitude of each object. The results from the Haversine Formula was compared with the distance *d* reported by the MobilEye and the differences were reported as an error. A total number of 145 objects were computed and the errors are represented in Figure 5. The plot shows that for all measured distances, the error magnitude is 10^{-10}. Even though the computed error has a low value, it is not possible to assume the object's location based only on this calculation. This is due to the possibility that the object's position, represented by its latitude and longitude, might be over a circumference of radius *d*, and centered on the vehicle's ref. point.

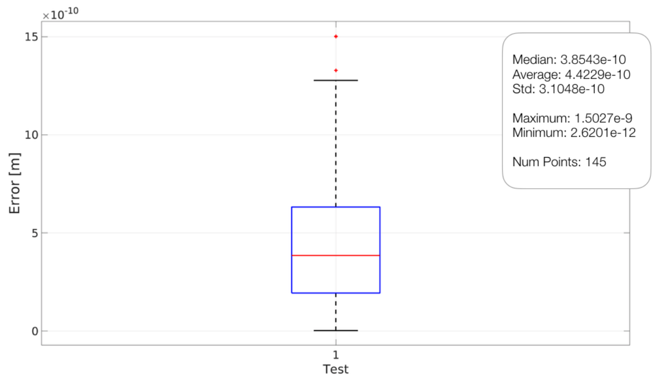

Fig. 5. Boxplot for the distance difference from the vehicle and the object measured by MobilEye and computed by GDO

Secondly, a test to measure the precision of the latitude and longitude reported by GDO was executed. This method employed a database with 25 points marked on the ground, with known latitude and longitude. The test vehicle was parked 14.33 meters apart from the marked area, and a pedestrian walked from one marked point to the other. Whenever the pedestrian stopped by a point, GDO calculated his geolocation. Figure 6 shows in red the distance between the vehicle's reference point and each point in database. The blue line is the value computed by the system. The mismatch between the lines represents the inaccuracy in meters calculated by the GDO.

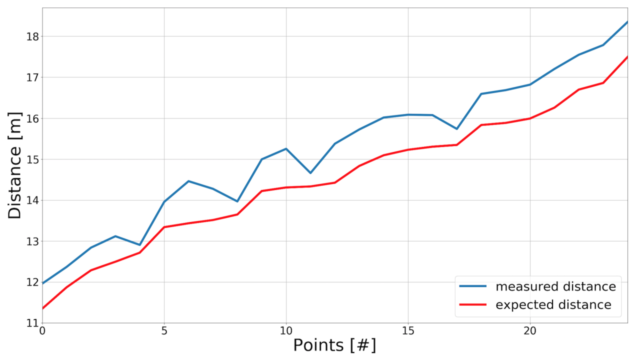

Fig. 6. Comparison between the expected distance and the measured distance

Afterward, the distances between the values presented in the database, and those reported by GDO were measured by applying the Haversine Formula. Figure 7 shows the errors computed by this distance. The lowest error is approximately 0.19 meters and is represented by the bottom line in the graphic. The dashed line connecting the lowest value to the bottom of the blue box symbolizes the first quartile (Q_1), i.e. 25% of the values. 50% of all values are between 0.6 and 0.9 meters - blue box, representing the second (Q_2) and third (Q_3) quartiles. The red line inside the box is the median, which represents an error of 0.80 meters. The dashed line between the top of the box and the highest horizontal line is the fourth quartile (Q_4), meaning that 25% of the values are between 0.9 and 1.03 meters. The highest error computed in this test is 1.03 meters, and is represented in the figure by the horizontal top line. The average value for this test is 0.74 meter with a standard deviation of 0.22 meters. Furthermore, it is observed that the values are more concentrated closer to the highest error than to the lowest value.

Hereafter, the GNSS representing the position of the test vehicle and GDO has been gathered to the OpenStreetMap [19] to implement a visual tool to simulate the Dynamic Map, which is depicted in Figure 8. The figure is divided into two parts, wherein the left side shows the map with three blue dots that are the test vehicle, a pedestrian, and a parked car. At the same time, the right side shows the cameras' output. The detected objects and reported on the map are delimited by a green polygon.

VI. Conclusion and Future Work

In concordance with the requirements established in Section II, it has been shown that employing GDO assembled

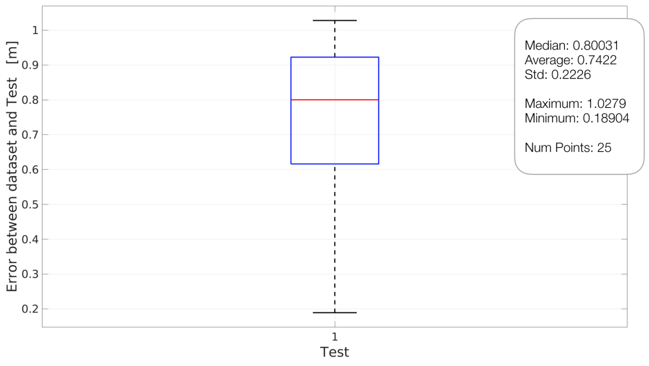

Fig. 7. Boxplot for the distance between the points reported in the data-set and the position computed by GDO

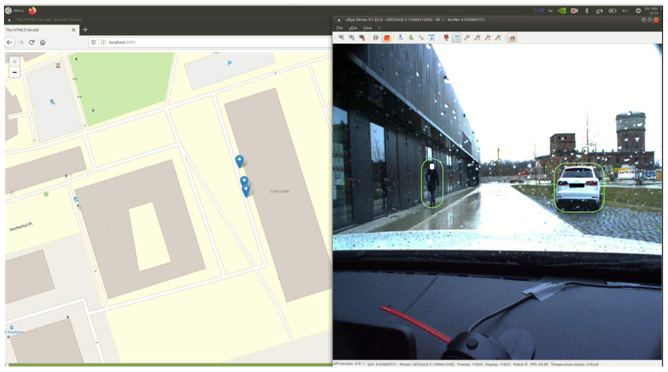

Fig. 8. Simulation of the Dynamic Map

with MobilEye Camera as a vision-sensor, it is feasible to compute the geolocation of detected objects with errors that might impact environment safeness. Additionally, GDO was developed to be suitable for any sensor attached to the vehicle, and from the CPM point of view, GDO assists in the development of a DM, computing objects' geolocation. Intending to evaluate GDO's accuracy and reliability, the distance between the detected object and the vehicle's reference point, reported by the MobilEye, has been compared with the distance calculated, employing the Haversine Formula, between the same points, using their latitude and longitude. A second validation method was implemented considering a database with 25 points, where latitude and longitude for each point were known. A pedestrian walked from one point to another, and his geolocation was computed by the GDO. Afterward, for each point, the Haversine Formula was applied to identify the distance between the geoposition reported in the database and the value calculated by the GDO. The data analysis of the second method showed an average error of 0.74 meters with a standard deviation of 0.22 meters. This error might be due to the size of the human body, which is much larger than the point measured by the GNSS. An average value for the Earth's radius has been considered and not the value corresponding to the position where the tests were implemented. Although this research has been based on the CPM requirements, the geoposition from objects detected by the vehicle have been merely computed. Thus, future works to analyze the error, intending to reduce it, and to measure its influence over the safety would be required. Furthermore, this algorithm can be gathered within Vanetza[5] to broadcast the CPM and define objects' geolocation at the receiver side according to VANETs' parameters.

Acknowledgments

The authors would like to thanks Andreas Festag, Quentin Delooz and Christina Obermaier for their support, Matheus Zimmermann for the GNSS database, and all members of the Car2X Lab at Technische Hochschule Ingolstadt.

References

[1] A. Festag, "Cooperative intelligent transport systems standards in europe," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 166–172, 2014.

[2] C. Sommer and F. Dressler, *Vehicular Networking*. Cambridge University Press, 2014.

[3] S. E. et al., *Triggering Conditions and Data Quality Adverse Weather Conditions*, September 2019.

[4] T. Biehle and K. Krumbiegel, *Triggering Conditions and Data Quality Dangerous Situation*, September 2019.

[5] J. B. et al., *Triggering Conditions and Data Quality Exchange of IRCs*, September 2019.

[6] S. E. et al., *Triggering Conditions and Data Quality Stationary Vehicle Warning*, September 2019.

[7] European Telecommunications Standards Institute (ETSI), *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Analysis of the Collective Perception Service (CPS); Release 2*, December 2019.

[8] F. A. Schiegg, N. Brahmi, and I. Llatser, "Analytical performance evaluation of the collective perception service in c-v2x mode 4 networks," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pp. 181–188, 2019.

[9] B. Mourllion, A. Lambert, D. Gruyer, and D. Aubert, "Collaborative perception for collision avoidance," in *IEEE International Conference on Networking, Sensing and Control, 2004*, vol. 2, pp. 880–885 Vol.2, 2004.

[10] H. Günther, R. Riebl, L. Wolf, and C. Facchi, "Collective perception and decentralized congestion control in vehicular ad-hoc networks," in *2016 IEEE Vehicular Networking Conference (VNC)*, pp. 1–8, 2016.

[11] H. Günther, B. Mennenga, O. Trauer, R. Riebl, and L. Wolf, "Realizing collective perception in a vehicle," in *2016 IEEE Vehicular Networking Conference (VNC)*, pp. 1–8, 2016.

[12] V. Kumar and V. Anand, "Development of web map service for openstreetmaps (osm) data," in *Proceedings of International Conference on ICT for Sustainable Development*, vol. 1, pp. 309–318, July 2015.

[13] U-Blox, "U-blox M8 Receiver description including protocol specification." [online], March 2020. `https://www.u-blox.com/sites/default/files/products/documents/u-blox8-M8_ReceiverDescrProtSpec_\%28UBX-13003221\%29.pdf`, last viewed April 2020.

[14] VARIENSE INC., "Inertial Measurement Unit VMU931." [online], March 2018. `http://variense.com/Docs/VMU931/VMU931_UserGuide.pdf`, last viewed March 2020.

[15] F. D. Christoph Sommer, *Vehicular Networking*. Cambridge University Press, 2014.

[16] A. Katriniok, *Optimal vehicle dynamics control and state estimation for a low cost GNSS-based collision avoidance system*. PhD thesis, RWTH Aachen University, 2014.

[17] T. Vincenty, "Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations," *Survey Review*, vol. 23, no. 176, pp. 88–93, 1975.

[18] H. Alkan and H. Celebi, "The implementation of positioning system with trilateration of haversine distance," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pp. 1–6, 2019.

[19] OpenStreetMap, "Openstreetmap." [online]. `https://openstreetmap.org`, last viewed March 2020.

[5]http://www.vanetza.org