Topological road mapping for autonomous driving applications

Esther Murciego[1], Carlos G. Huélamo[2], Rafael Barea[2], Luis M. Bergasa[3], Eduardo Romera[2], Juan F. Arango[2], Miguel Tradacete[2], Álvaro Sáez[2]

[1] Deparment of Electronics, University of Alcala, Spain,
esther.murciego@edu.uah.es
[2] Member of RobeSafe researching group,
Deparment of Electronics, University of Alcala, Spain,
carlos.gomezh@edu.uah.es
rafael.barea@uah.es,
eduardo.romera@uah.es jfelipe.arango@depeca.uah.es
tradacete.miguel@gmail.com alvaro.saezc@edu.uah.es
[3] Head of RobeSafe researching group,
Deparment of Electronics, University of Alcala, Spain,
luism.bergasa@uah.es,
www.robesafe.es/personal/bergasa/

Abstract. We present a system to carry out the elaboration of efficient topological maps for autonomous driving applications by using lanelets in satellite images provided by an open source tool such as JOSM (Java Open Street Maps). Our approach is motivated by the need for more efficient and frequent updates on large-scale maps in self driving applications. The system chains the elaboration of an adjusted topological map by using JOSM and its subsequent evaluation using both V-REP simulator and real world tests based on a route planning method (Dijkstra) and trajectory tracking algorithms to implement a ROS based autonomous electric vehicle able to drive in urban areas.

Keywords: Lanelets, Open Street Maps (OSM), Java OSM (JOSM), Dijkstra, autonomous Driving

1 Introduction

Over the past few years, a topic that focuses the attention of automotive companies is the development of autonomous cars. Statistic show that over 70 % of the population in the European Union (EU) is living in urban areas. According to the World Health Organization, nearly one third of the world population will live in cities by 2030. Aware of this problem, the Transport White Paper published by the European Commission in 2011 indicated that new forms of mobility should be proposed to provide sustainable solutions for people and goods safely. Regarding safety, it sets the ambitious goal of halving the overall number of road deaths in the EU between 2010-2020. However, the goal will not be easy, only in 2014 more than 25,700 people died on the roads in the EU (18 % reduction).

Autonomous driving is considered as one of the solutions to the before mentioned problems and one of the great challenges of the automotive industry today. The existence of reliable and economically affordable autonomous vehicles will create a huge impact on society affecting environmental, demographic, social and economic aspects. In particular, it is estimated to cause a reduction in road deaths, improved traffic flow, reduced fuel consumption and harmful emissions associated, as well as an improvement in the overall driver comfort and mobility in groups with impaired faculties, like the elderly or disabled people. Autonomous driving has attracted much attention recently by the research groups and industry, due to the billboards of various companies on expectations of market entry. However, his predictions seem to be very optimistic. A more scientific organization, such as IEEE, has recently predicted that by 2040 the majority of vehicles traveling on highways will be autonomous. Driving in urban environments will take longer, due to its complexity and uncertainty.

The autonomous car must be able to navigate without making mistakes, consequently it has to understand the environment. For this proposal, apart from using the most advanced techniques of sensory perception that recognizes the environment in a robust way, it is pretty useful to use a map with topological information of places the car is supposed to be driven.

In this work we address the problem of efficient elaboration and maintenance of topological maps for autonomous driving applications, using an open source map format, easy to integrate and merge with other maps such as OpenStreetMaps [1]. We model relevant parts of the conductive zones with lanelet elements, which are atomic, interconnected and manageable road segments, geometrically represented by a left and right link. Forming a network of nodes with the topological information of the terrain.
The lanelet will not only establish the road and its limits to follow, it will have integrated, the road behaviour that must follow the car too.
The topological mapping approach based on lanelets has been validated in simulation cases using our navigation architecture made in ROS [2] and V-REP simulator [3] [4] and our open-source electric vehicle for the real world tests.

2 Open Street Maps

Open Street Maps (OSM) is a collaborative project with the purpose of implementing free and editable maps. These maps are generated by using geographic information caught by mobile GPS, ortoimages and other open-source tools, all this information distributed under Open Database License (ODbL). Registered users can upload their contributions by GPS devices and can introduce or correct vectorial data through the use of edition tools created by OpenStreetMap community. In this work we focus on Java OpenStreetMap editor (JOSM), often used by more experienced OSM contributors when a large dataset must be introduced.

2.1 Java Open Street Maps

JOSM is an editing tool with a similar interface to other traditional geographical information systems (GIS) packages. This application allows the user to edit, tag or import

OSM data offline, not being required online connection. For that reason, people all over the world can edit the same zone at the same time through the use of JOSM and then uploading this new data to Open Street Map through the OSM application programming interface (API), being JOSM responsible to merge changes if several people upload different data of the same area. On the other hand, additional services such as linking OSM features, images, audio notes or supporting data conflict resolutions are provided. Moreover, this tool can be improved by using plug-ins, such as RoadSigns and Measurements plug-ins which were set in order to locate different traffic signs related to regulatory elements appropriately.

Main mapping elements in JOSM are:

- **Node:** Point that refers to a given geographical position.
- **Way:** Ordered node list that represents a polyline or polygon (if the polyline starts and ends in the same point).
- **Relation:** Group of nodes, paths or other relations with common properties.
- **Tag:** Fields in which a relation is segmented. Tag sintaxis is divided into a key, a feature and its value. Example: speedlimit = 40 defines that this section has 40 km/h as speed limit.

2.2 Lanelets

When creating an efficient map, both in geometric and topological aspects, an accurate description of its elements as well as their relations among them must be supported. Different approaches so as to create this efficient map are:

- **Geometric approach:** Geometric map of the environment.
- **Topological approach:** Abstract description of the map, including the different relationships among its members.
- **Geometric and topological approach:** Merging above approaches.

For that reason, we choose a relatively novel concept called lanelets [5]. This method was proposed originally by Die Freie Universitt Berlin with the purpose of drawing an efficient modifying, creating and maintenance map according to a mapping technique able to merge both geometric and topological approach.

A lanelet is defined as an atomic lane segment featured by its left and right bound. These bounds are polylines, and therefore allow for arbitrary precise approximation of lane geometries. The role of a bound, both left and right, specifies the driving direction.

In order to enable routing through a map, a graph made up by adjacent lanelets must be generated. This one can be carried out in two ways: Let the polyline be located in the center of the road or create two polylines that delimit a lane in which the car must be kept, circumscribing the driving zone.

However, if the first option is chosen, a problem arises: The car does not take into account the lane limits, crucial error in autonomous driving applications. For that reason, throughout this project the external campus of the University of Alcala was generated by using JOSM tool with two polylines per each lanelet, each one representing both left and right bound, as illustrated in Figure 1.

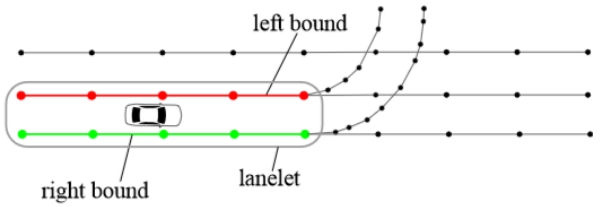


Fig. 1. Map composition by using lanelets. Notice that each lanelet is made up by a left and right bound.

2.3 Mapping techniques

JOSM offers different choices so as to map our Campus, such as Bing aerial image, Mapbox satellite image, Spain Cadastre or Spain satellite image, among others. For our purposes, since the autonomous car requires an accurately precision, the Spain satellite image, also called PNOA, was used due to its high resolution.

The Plan Nacional de Ortofotografía Aérea (PNOA) has as aim getting aerial ortoimages with resolution from 25 to 50 cm and high precision digital models of elevation of the whole Spanish territory, with an update period spanning from two to three years, depending on the zones. It is a cooperative project cofinanced by General Administration of the State and Autonomous Comunities. Figure 2 shows the JOSM interface and our manually two dimensions mapped external campus of the University of Alcala based on PNOA satellite image. In an easy way both lanelets, regulatory elements and different stations can be added or modified. Figure 2 shows from bottom to top several windows were relationships are shown, and the same one to layers and members. Notice that though the format file is OSM, these data are structured in an XML format which provides great portability and easy access to the different members of the map.

However, it is pretty often that the JOSM tool shows a misaligned aerial image, representing a hazardous matter if we pretend to define clear and accurate lanelets where the autonomous car must be driven. In order to solve this problem, we introduce a contribution based on multi-DGNSS positioning integrated in the car with the purpose of verifying the accurate of this satellite image.

The main module of the localization systems consists of a multi-constellation system (multi-GNSS) with RTK positioning solution. This module is directly integrated in the back of the car, made up by two elements: Differential Hiper Pro GPS + receiver configured as rover and a local base station with the purpose of generating differential corrections for the rover.

The rover can obtain data from both GPS and GLONASS to provide a more robust solution than standard GPS by increasing the number of visible satellites. It is able to provide information at a frequency of 10 Hz through the use of differential corrections in order to improve the required accuracy, since autonomous vehicles demand real-time information of the surroundings.

In addition, we set a local based station based on Choke-Ring Antenna, specifically chosen to deal with multipath, connected to a second Hiper Pro GPS + receiver that pro-

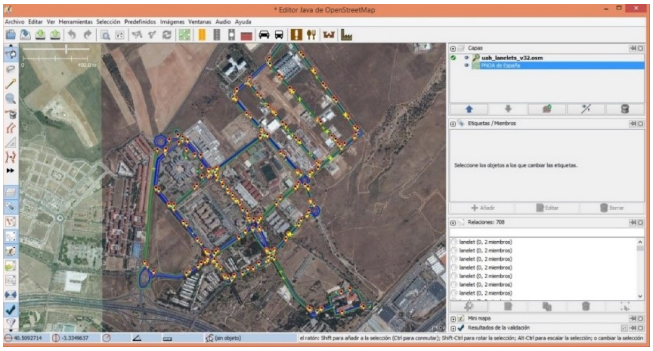


Fig. 2. JOSM interface. It can be observed the external campus of the University of Alcala manually mapped by using lanelets technique

vides these differential corrections. As mentioned, these corrections are generated at 10 Hz, published over Internet by using standard open source software in the autonomous vehicle using a GPRS link via radio.

With these differential corrections working, we mapped those zones which were not updated in the PNOA satellite image or even checked, if required, those zones whose drawn lanelets were not matched to these differential corrections projected into JOSM.

Moreover, several fields useful for our ROS based autonomous vehicle architecture were tagged, both for simulation and real world, being found in all lanelets in order to take into account in an easier way the behaviour that the car must carry out. The lanelet fields are made up by:

- **Tags:**
  - **Layer:** Set of members to which a lanelet belongs within the map.
  - **Speedlimit:** Speed to which a road is limited within the map.
  - **Type:** Kind of each member. In this work it is defined as lanelet, station or regulatory element.
  - **Role:** Tag that models the behaviour of a given lanelet. In this workd it is defined as merging, split, intersection or roundabout.

- **Members:** Elements that make up a lanelet.
  - **Polylines:** Geometric limits that define both left and right bound of a lanelet.
  - **Other relations:** Regulatory elements or stations.
  - **Node roundabout:** Node required if a lanelet is featured with a role roundabout. It can be found in the center of the roundabout.

Regulatory elements were defined as followed:

- **Tags:**
  - **Type:** Regulatory element.
  - **Maneuver:** Behaviour carried out when this regulatory element appears:
    - Give way (give_way)

    - Pedestrian crossing (pedestrian_crossing)
    - STOP
    - Traffic light
  - **Layer:** Set of members to which the regulatory element belongs within the map.

- **Members:** Elements that make up a regulatory element.
  - **Stop line:** Topological line that sets the limit at which the car should stop.
  - **Reference (ref):** Node that represents the traffic signal of a given regulatory element. If it is a traffic light, it will have more than one reference point: the reference point of its own traffic light and the other reference points of those traffic lights with which the first one is synchronized.
  - **Activate Point (activate_point):** Point in which the car is warned that a few meters from it, there is going to be a signal reference.
  - **End point (end_point):** Point located a little farther than the stop line indicating the maximum point that can be reached if for some reason the stop line is exceeded. It is only defined in semaphores.

- **Stations:** Possible stopping points for the car.
  - **Tags:**
    - **Type:** In this case it is equivalent to 'station'.
    - **Name:** Name of the station. There are 27 stations in our map.

3 Route planning

So as to obtain an accurate route planning, first a self-code algorithm for testing lanelets connectivity must be run. Both connectivity checking and Dijkstra algorithm are based on "liblanelet" library [6], available online.

This connectivity is used to check if the relationships among lanelets are correct. The code goes through the nodes of the lanelets and focuses on those that are located at the beginning and at the end of each lanelet. The code analyzes which is the lanelet to which a given node is the end and which is the lanelet to which the same node is the beginning, returning as output that both lanelets are connected.

By using this method it can be checked that all lanelets have at least two connections. This quickly detects which and where are located wrong lanelets, since, if exist wrong connections, the subsequent routing algorithm can not be executed.

In our map all lanelets are connected except for the Polytechnic School. This building has both an absolute beginning node and an absolute ending node, that is, only are related with a single lanelet.

On the other hand, despite of topological and geometrical features of previous lanelets, routes calculation and navigation trajectories are not developed by using JOSM since there are regions that have not got enough data to be reliable. However, there are some searching algorithms based on graphs, such as Dijkstra algorithm [7]. For our purpose we chose the Dijkstra method or minimum path algorithm for its efficiency and speed in our working environment.

This algorithm determines the shortest path, given a vertex of origin to the rest of vertices. The graph is made up by nodes separated from each other by different ways and each one with its own name. Depending on the lanelet length a weight is assigned to each one, obtaining a weighted graph required as input to be useful for this algorithm, as illustrated in Figure 3. The longer a lanelet is, the more it will weight. Each stop has its own coordinates and through the use of the Euclidean distance we get the closest lanelets, which are used as an argument to get the shortest route with the function shortest_path, of the libLanelet library.

[Distance, Predecessor Node] (Iterations)

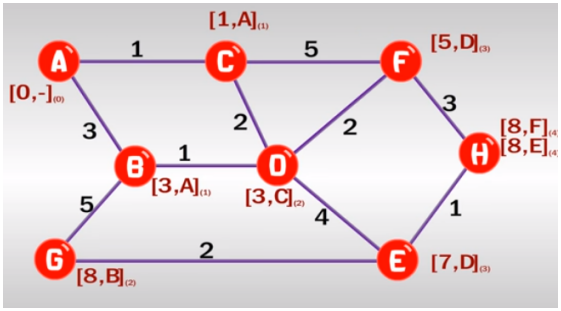


Fig. 3. Dijkstra applied in a graph

4 Simulation enviroment

Our map was validated by using both RVIZ [8], simulator that belongs to ROS, in order to monitorize the topics and specify the route, and V-REP [3], property of Coppelia and compatible with ROS, simulating the different behaviours cases, such as pedestrian crossing, stop or given way, apart from simulating the vehicle dynamics properly.

V-REP allows importing mesh formats, so an OBJ file is required. Therefore, our map, that is in OSM format, was transformed into OBJ format by using an open source tool called OSM2World.

The visual part of the car seen in simulation has been designed according to our real prototype (Figure 4), as we show in Figure 5), imported from a CAD repository adjusting all parameters to this real prototype. It is made up by different car parts as: four wheels with cushioning, propulsion engine, brakes, steering, sensors and a curve base structure based on cuboids.

To obtain a more realistic simulation, the vehicle steering system was simulated in V-REP as a dynamic system, fed up by a servomotor controlled in position. The steering wheel position is internally controlled by V-REP using a PID controller.

In addition, we model different sensors carried by the vehicle, with the purpose of perceiving the environment, such as a LiDAR, a GPS and stereo camera.

Once a map based on lanelets is provided, we use RVIZ simulator, connected to VREP by code, in order to specify the start and goal point of our route, parameters which can not be defined in V-REP. After defining in RVIZ a start and goal point, a series of equidistant points are generated in the centreline of the respective lanelet and the purepursuit algorithm [9] is used to follow this discretized trajectory. Two navigation ways (Figure 6) can be executed to follow this path:

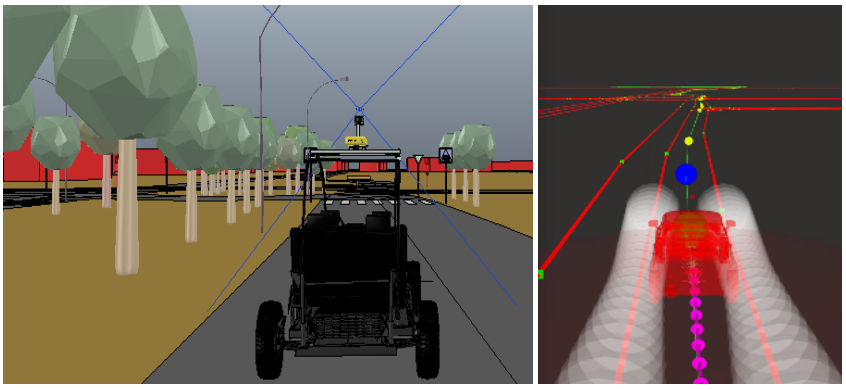
Fig. 4. Our real prototype equipped with sensors

- **Non-reactive navigation:** Navigation way that only executes the tracking algorithm, without taking into account neither lanelets boundaries nor behaviour cases, such us pedestrian-crossing, give way or stop. It describes a constant radius that causes the vehicle moves to the next target point. The lookahead is adjusted in order to decrease the car speed if the car detect great curvatures, just like a human driver would do.
- **Reactive navegation:** This navigation way takes into account previous laws of Non-reactive navigation but furthermore it is able to face and react both to lanelets boundaries and obstacles, such as pedestrians or vehicles, both in static and dynamic, by using the Curvature-Velocity Method (CVM) [10] and precision tracking algorithms [11], drawing arcs of dynamic radius that describe the possible trajectories to reach the target point avoiding the ahead obstacles.

5 Results

According to these two navigation ways, non-reactive and reactive, we designed a series of routes with the purpose of validating our method, first in simulation and second

with a real prototype. Table 1 shows different routes all over the external campus of the University of Alcala with a moderate degree of complexity presenting streets with different speeds, curves and various traffic signals. Each route presents a number of lanelets related to the total distance. The average distance for a lanelet was 116.125 metres.



(a) V-REP simulator (b) RVIZ simulator

Fig. 5. Figure (a) shows our imported car in simulation by using V-REP simulator. Figure (b) shows route planning in R-VIZ simulator

Table 1. Routes used in simulation

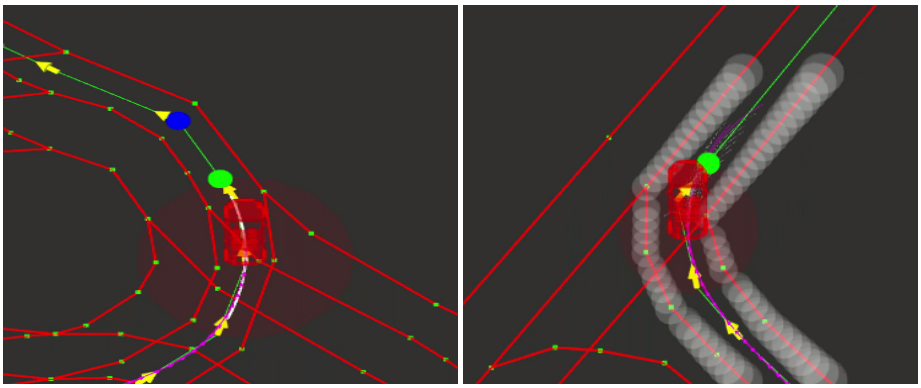
Route	StartPoint	EndPoint	Metres	N lanelets
1	PolytechnicStart	Train Station	1800	20
2	Train Station	PolytechnicEnd	2500	32
3	Biology	University Hospital	1700	8
4	Universitary Residence	CGD Telefnica	2500	25
5	Pharmacy	Polytechnic	1800	18

On the other hand, Table 2 shows the number of roundabouts per each route, its average speed, total time of the route and its respecting navigation way. It can be observed that in spite of the fact that the route 2 and 4 are longer than route 1, because route 1 presents more curves and reactive control is taken into account, it takes more time driving the car throughout the simulation, the same than in real tests. In conclusion, the bigger the amount of curves taking into account reactive control, the more time it will take driving the simulation.

As example, it is explained the route 1, which starts at the Polytechnic school and ends at the Train Station.

The stations and the route of lanelets followed is: *Startstation: PolytechnicStart Endstation: Train Station*
Route: [<Lanelet -51855 >, <Lanelet -51851>, <Lanelet -51837>, <Lanelet -51051>, <Lanelet -51221>, <Lanelet -51087>, <Lanelet -51083>, <Lanelet -51085>, <Lanelet -51217>, <Lanelet -51069>, <Lanelet -51139>, <Lanelet -51149>, <Lanelet -51203>, <Lanelet -51205>, <Lanelet -51451>, <Lanelet -51457>, <Lanelet -51475>, <Lanelet -51561>, <Lanelet -51531>, <Lanelet -51533>]

Figure 7 shows the simulation, both in V-REP and RVIZ, under reactive navigation, at the beginning of the route 1. It can be seen the curvature lines in fuchsia color, target points in blue and the path to follow in green.



(a) Non-Reactive Navigation (b) Reactive Navigation

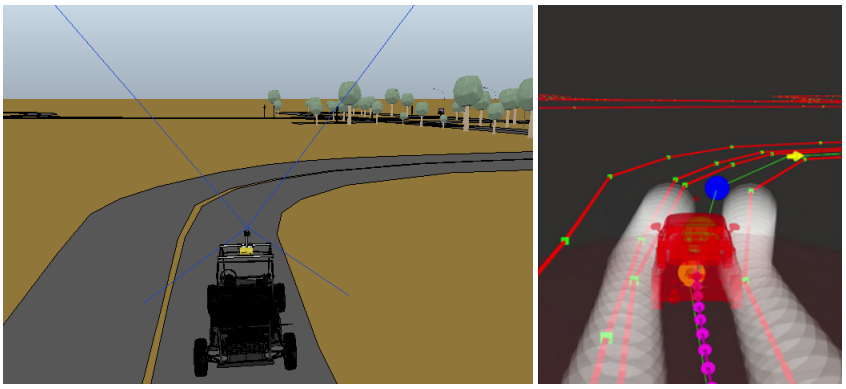
Fig. 6. Figure (a) shows non-reactive navigation and Figure (b) shows reactive navigation

Table 2. Quantitative parameters of routes

Route	Roundabouts	Average speed (Km/h)	Time (Seg)	Control
1	4	22.6	491	Reactive
2	5	23.4	266	No reactive
3	4	21.9	290	No reactive
4	7	23.7	284	No reactive
5	5	20.8	208	No reactive

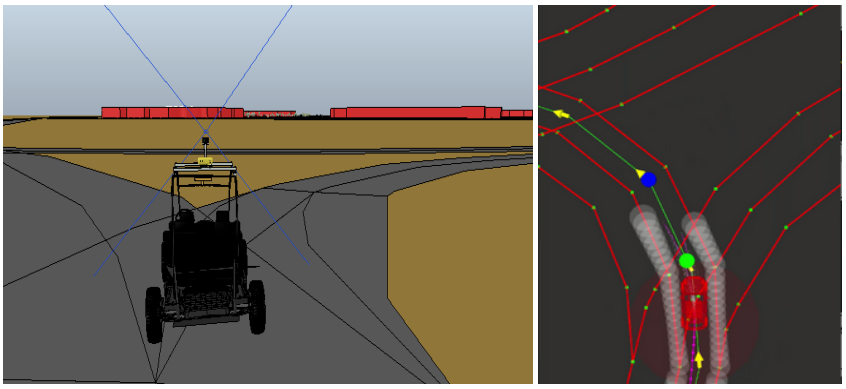
Figure 8 shows, from above, in one of the roundabouts of the route. The blue target points are observed and the lanelets are detected as obstacles. In figure 9, it can seen the route that the vehicle has followed throughout the Campus, using the simulator. Since we don't see the Campus image as background in Figure 9, we show the followed route most clearly in Figure 10. As supposed, the shortest route has been chosen. This route

has been simulated using the reactive navigation, whose processing time is greater that the non-reactive because non-reactive doesn't have to calculate the obstacles that appear in the route to define its path and its speed. In this case the duration of the route is about 8 minutes and 11 seconds, as illustrated in Table 2.



(a) V-REP simulator (b) RVIZ simulator

Fig. 7. Back view on the route Polytechnic school - Train Station.



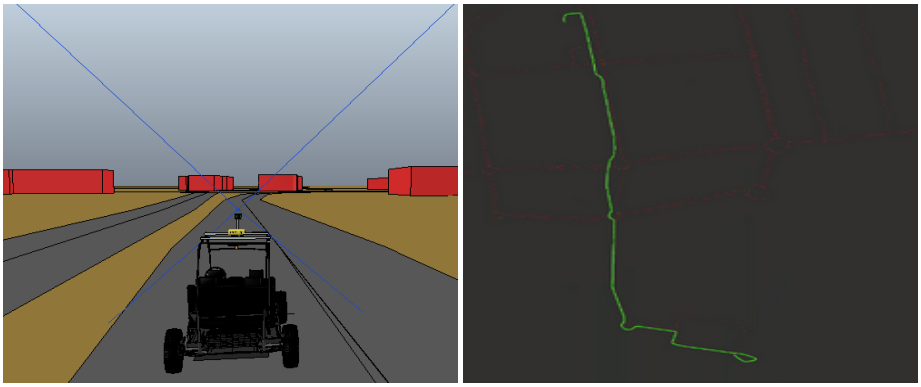
(a) V-REP simulator (b) RVIZ simulator

Fig. 8. Roundabout situation in the route Polytechnic school - Train Station.

Finally, Figure 11 shows different cases both in simulation and with our real prototype following the route Polytechnic school - CGID Telefonica.

6 Conclusions

The implementation of the lanelets map based on the external campus of the University of Alcala has contributed to the final objective of the SmartElderlyCar project [4] which consists of the development of an electric vehicle capable of driving autonomously



(a) a (b) b

Fig. 9. Global view of the driven route 1 by using V-REP

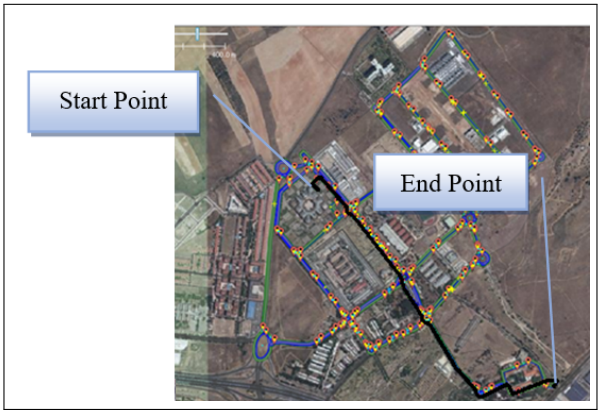


Fig. 10. JOSM view of the route 1: Polytechnic school to Train Station

by our campus. The OSM map doesn't provide enough data to achieve autonomous driving, so it has to be enriched by lanelets that delimit the lane with two polylines thus preventing the car from moving outside the limits established by the road.
In addition, with the method of lanelets it can be included in the same map regulatory elements for the car. The proposal has been validated in simulation using V-REP under ROS by two navigation methods (reactive and non-reactive) and in real tests obtaining successful results.

7 Acknowledgment

This work has been partially funded by the Spanish MINECO/FEDER through the SmartElderlyCar project (TRA2015-70501-C2-1-R), the DGT through the SERMON

project (SPIP2017-02305), and from the RoboCity2030-III-CM project (Robótica aplicada a la mejora de la calidad de vida de los ciudadanos, fase III; S2013/MIT-2748), funded by Programas de actividades I+D (CAM) and cofunded by EU Structural Funds.

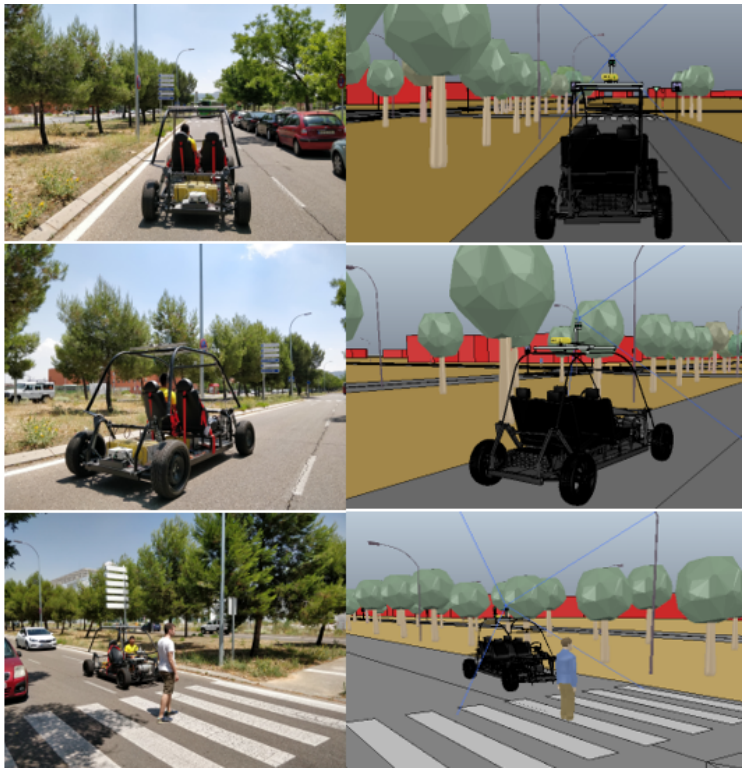


Fig. 11. Different situations found in the route 1, both with our real prototype (left column) and in simulation (right column)

References

1. M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *Ieee Pervas Comput*, vol. 7, no. 4, pp. 12–18, 2008.
2. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.
3. E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pp. 1321–1326, IEEE, 2013.
4. C. Otero, E. Paz, J. López, R. Barea, E. Romera, E. Molinos, R. Arroyo, L. M. Bergasa, and E. López, "Simulación de vehículos autónomos usando v-rep bajo ros," *Actas de las XXXVIII Jornadas de Automática*, 2017.

5. P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pp. 420–425, IEEE, 2014.
6. J. Högdahl, "Design specification autonomous trucks," 2014.
7. U. Brandes, "A faster algorithm for betweenness centrality," *Journal of mathematical sociology*, vol. 25, no. 2, pp. 163–177, 2001.
8. D. Gossow, A. Leeper, D. Hershberger, and M. Ciocarlie, "Interactive markers: 3-d user interfaces for ros applications [ros topics]," *IEEE Robotics & Automation Magazine*, vol. 18, no. 4, pp. 14–15, 2011.
9. R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," tech. rep., Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, 1992.
10. R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, vol. 4, pp. 3375–3382, IEEE, 1996.
11. E. Oron, A. Kumar, and Y. Bar-Shalom, "Precision tracking with segmentation for imaging sensors," *IEEE Transactions on aerospace and electronic systems*, vol. 29, no. 3, pp. 977–987, 1993.