Hindawi
Journal of Advanced Transportation
Volume 2018, Article ID 6392697, 10 pages
https://doi.org/10.1155/2018/6392697

Research Article

Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles

Pablo Marin-Plaza, Ahmed Hussein, David Martin, and Arturo de la Escalera

Universidad Carlos III de Madrid, Leganes, Spain

Correspondence should be addressed to Pablo Marin-Plaza; pamarinp@ing.uc3m.es

Received 17 November 2017; Revised 16 January 2018; Accepted 17 January 2018; Published 22 February 2018

Academic Editor: Aboelmaged Noureldin

The aim of this work is to integrate and analyze the performance of a path planning method based on Time Elastic Bands (TEB) in real research platform based on Ackermann model. Moreover, it will be proved that all modules related to the navigation can coexist and work together to achieve the goal point without any collision. The study is done by analyzing the trajectory generated from global and local planners. The software prototyping tool is Robot Operating System (ROS) from Open Source Robotics Foundation and the research platform is the iCab (Intelligent Campus Automobile) from University Carlos III. This work has been validated from a test inside the campus where the iCab has performed the navigation between the starting point and the goal point without any collision. During the experiment, we proved the low sensitivity of the TEB method to variations of the vehicle model configuration and constraints.

1. Introduction

Autonomous vehicles base their decisions on planner modules that create the collision-free waypoints in the path to reach the destination point. These modules are capable of finding the optimal solution minimizing the computational time and distance covered by the vehicle avoiding static and dynamic obstacles. Some research groups test these modules on real vehicles such as PROUD in Parma in 2013 [1] or Karlsruhe Institute of Technology (KIT) driving around the cities with advanced technology in perception, navigation, and decision making [2]. In all cases, the navigation module is divided into a global planner and a local planner, where the first one finds the optimal path with a prior knowledge of the environment and static obstacles, and the second one recalculates the path to avoid dynamic obstacles.

The main contribution of this work is to demonstrate the integration of using TEB local planner in an Ackermann model with real test. The platform used is the iCab (Intelligent Campus Automobile) which works with the software prototyping tool ROS. Additionally, it has been discovered that the capacity of the vehicle to follow the local plan with accuracy is not mandatory. The vehicle is able to reach the goal with modest errors when following the local plan. This characteristic is really important in real life when the vehicle model depends on multiple factors as the pressure of the tires, the number of passengers, and the battery of the vehicle. When this mathematical model varies from the nominal values, the vehicle will not reach point by point the provided local plan, so TEB local planner is more robust to face changes in the configuration of the mathematical model.

In order to prove that the TEB local planner is able to work along the other processes involved in the vehicle, it is necessary to solve several problems related to the localization in the environment (initial localization), the ego-motion of the vehicle (odometry), environment understanding (perception and mapping), which path should the vehicle take (planning), and movement (low level control). Hence, this document gathers all the solutions for each task for navigation and then tests the Dijkstra method for global planning and the Time Elastic Bands method used for local planning.

This document is divided into six sections. Section 2 describes current planning methods used in the autonomous vehicles research field. Section 3 describes the research platform and the modules related to the navigation. Section 4 describes the experimental setup. Section 5 summarizes the results. Finally, the last section contains the conclusions and the future work.

2. Path Planning

The path planning problem is a well-known NP-hardness [3] where the complexity increases with the degrees of freedom of the vehicle. In this work, the ground vehicle is reduced to a 2D space of a single plane X-Y where the Z component is neglected and the vehicle is restricted by some constraints due to the Ackermann configuration [4]. The aim of solving this NP-hardness is not to find one solution that connects the start point and the goal point, but the optimal solution with the minimum distance and the smoothest maneuvers and without hitting any known obstacles. Usually, this module is divided into the global planner, which uses a priori information of the environment to create the best possible path, if any, and the local planner, which recalculates the initial plan to avoid possible dynamic obstacles. The generated plan is a collection of waypoints for global, $P_i = (x_i, y_i)^T$, and local $P_i = (x_i, y_i, \theta_i)^T, \in \Re^2 \times S^1$, where S is the navigation plane where the vehicle is able to move.

2.1. Global Planner. As said before, the global planner requires a map of the environment to calculate the best route. Depending on the analysis of the map, some methods are based on Roadmaps like Silhouette [5] proposed by Canny in 1987 or Voronoi [6] in 2007. Some of them solve the problem by assigning a value to each region of the roadmap in order to find the path with minimum cost. Some examples are the Dijkstra [7] algorithm, Best First [8], and A^* [9]. Another approach is by dividing the map into small regions (cells) called cell decomposition as mentioned in [10]. A similar approach by using potential fields is described in [11], the most extended algorithm used few years ago rapidly exploring random trees [12] or the new approach based on neural networks [13]. Some solutions combine the aforementioned algorithms improving the outcome at the cost of high computational power.

So, for this work and for testing purposes, the selected method is the Dijkstra algorithm because of the simplicity for debugging and its good performance on the experiments. It is necessary to remark that the goal of this paper is not to find the best and fastest solution, but it is to prove that all modules of navigation are working together and the vehicle is able to navigate from one point to another.

Dijkstra uses the roadmap approach to convert the problem into a graphic search method using the information of a grid cell map. This method starts with a set of candidate nodes where the vehicle is able to navigate (free space) assigning a cost value to each of them. From the starting point, this value is increased by the necessary number of nodes to pass through to reach each node. For example, in Figure 1, the free space around the starting point takes the value of one unit and for the second generation of neighbors takes the value of two units and so on. For each cell with value, a checked cell is assigned and the method continues with the next generation of neighbors until the goal point is reached. The minimum value of the sum of all nodes from the starting point and the end point is the shortest path. After the success of finding the global path from the start to the goal, all the selected nodes are translated into positions in the reference axes as the form $P_i = (x_i, y_i)^T$. The global planner divides the map into nodes for each free cell but the outcome is not smooth and some points are not compliant with the vehicle geometry and kinematics.

2.2. Local Planner. In order to transform the global path into suitable waypoints, the local planner creates new waypoints taking into consideration the dynamic obstacles and the vehicle constraints. So, to recalculate the path at a specific rate, the map is reduced to the surroundings of the vehicle and is updated as the vehicle is moving around. It is not possible to use the whole map because the sensors are unable to update the map in all regions and a large number of cells would raise the computational cost. Therefore, with the updated local map and the global waypoints, the local planning generates avoidance strategies for dynamic obstacles and tries to match the trajectory as much as possible to the provided waypoints from the global planner. Different approaches are based on trajectory generation using Clothoids lines [16], Bezier lines [17], arcs and segments [18], or splines [19]. All of them create intermediate waypoints following the generated trajectory. The selected method for local planning is Time Elastic Bands [15, 20] which consists of deforming the initial global plan considering the kinematic model of the vehicle and updating the local path based on dynamic obstacles or the possible deviation from the path. This method has been selected because of the integration of the trajectory planning and obstacle detection with avoidance at low computational cost fulfilling the requirements of the local planner in the research platform iCab.

Time Elastic Bands (TEB) create a sequence of intermediate vehicle poses $p_i = (xi, yi, \theta_i)^T$ modifying the initial global plan. It requires the velocity and acceleration limits of the vehicle, the security distance of the obstacles and the geometric, and kinematic and dynamic constraints of the vehicle. All of this configuration generates a set of commands for speed (v) and steering angle (δ), as $\mathrm{cmd}_i = (v_i, \delta_i)^T$, required to achieve the intermediate waypoints, while the vehicle is moving. Figure 2 shows a robot moving from the starting point to the goal point passing through four waypoints in the smoothest way possible between obstacles. One of the drawbacks of this method is the influence of the dynamic obstacles direction in the generation of the recalculated local path. For example, when a pedestrian is crossing from east to west the trajectory generated which connects a point from the south to the north (Figure 3), the recalculation of the new trajectory will still be on the left of the obstacle but further. The solution is the creation of three elastic bands: one updated each iteration, one is recalculated from the original path, and the last one is created from scratch each time. The selected one is the shortest of the three that is compliant with all requirements.

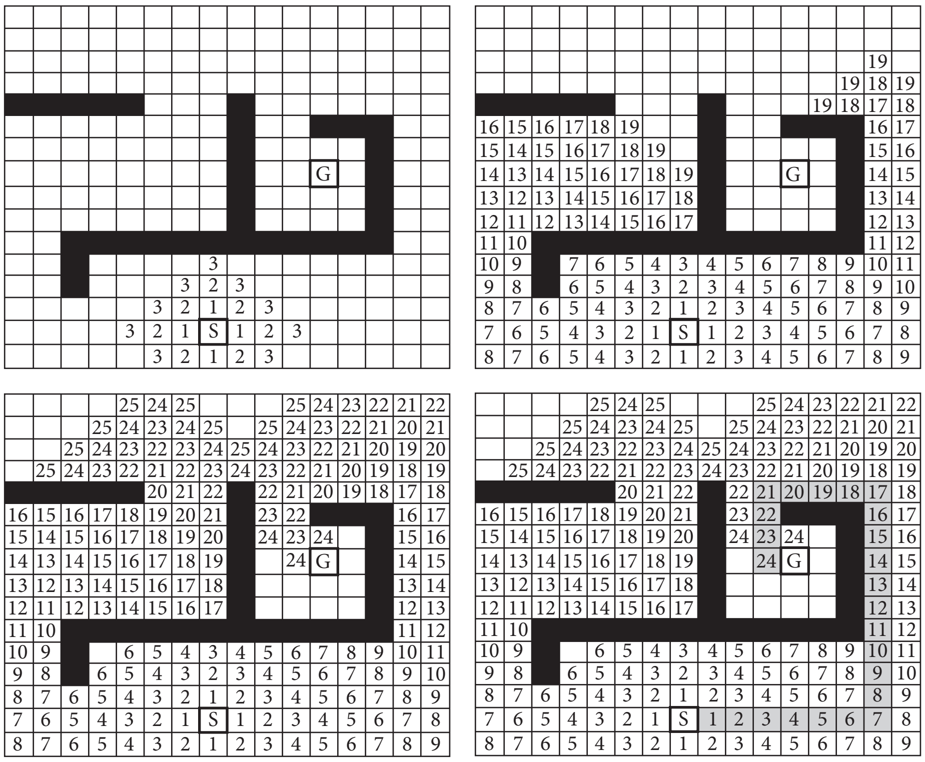


Figure 1: Dijkstra method to find the optimal path [14].

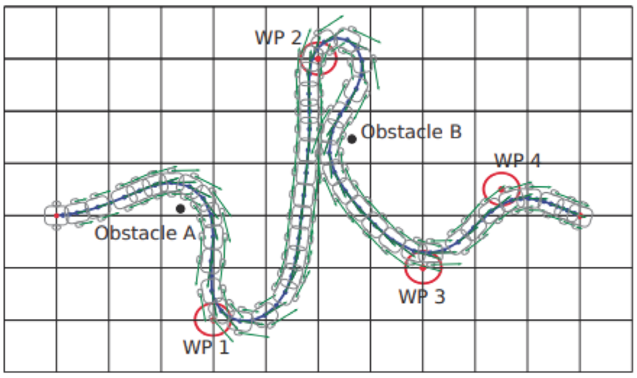


Figure 2: Description of Time Elastic Bands from [15].

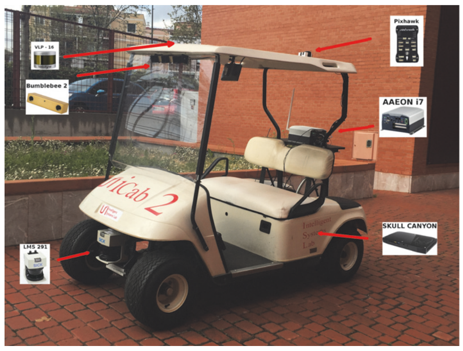


Figure 4: iCab with sensors.

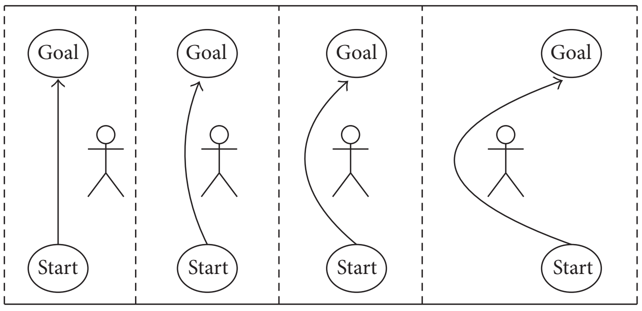


Figure 3: Pedestrian crossing from east to west of the trajectory generated.

3. Platform Description

The research platform is a modified golf cart equipped with several sensors such as wheel encoders, a Lidar with 16 laser planes on the top, a single plane laser located in the front, a stereo camera located in the front, and a GPS-IMU-Compass located on the top, as shown in Figure 4. The movement of the vehicle is done by a 36 V DC motor for the linear acceleration and another 36 V DC motor for the steering wheel. More information and tools are available in [21–23].

Additionally, the vehicle includes two computers to gather information from the sensors, process it, and perform the navigation decisions. The main computer is in charge of the wheel encoders, images, laser, imu, GPS, and compass

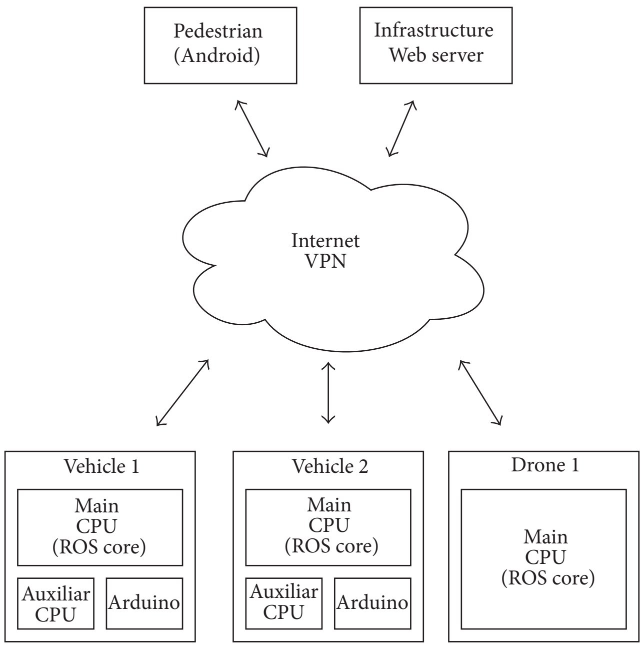


Figure 5: System communications.

and generates the maps. Therefore, the system provides wheel odometry, obstacle detection, and maps. The second computer processes the point cloud generated from the Lidar and computes the Lidar odometry. The software used, as aforementioned said, is ROS. This tool allows the vehicle to share and synchronize messages between nodes in the same computer and, additionally, with the computers and microcontrollers in the vehicle by the network using ROS core master. Furthermore, using a master discovery tool for all the ROS core masters in the network (one core master for each vehicle), the systems are able to share messages between vehicles, infrastructures, and pedestrians [24]. The use of one core master for each vehicle allows the inner communications even when there is no Internet connection, as shown in Figure 5.

A brief graphical description of the nodes and connections is shown in Figure 6. From left to right, the nodes are as follows:

(i) *Global map* is responsible for loading the global grid map for the campus as *navigation_msgs/OccupancyGrid.msg*. The selected grid size has a resolution of 0.156 m used in the analysis of the previously recorded point cloud obtained with the Lidar (a priori knowledge).

(ii) *Global planner* uses the Dijkstra algorithm to create the/*global_plan* to reach the goal point using the global map. This path message is a standard type *navigation_msgs/Path.msg* which contains the waypoints without the orientation.

(iii) *Lidar odometry* and *wheel odometry* are in charge of creating the ego-motion measurements from the starting point of the vehicle. The messages used are the type of *navigation_msgs/Odometry.msg*.

(iv) *Odometry manager* is responsible for the fusion and adding the initial localization for the conversion between global and local odometry.

(v) *Costmap 2D* uses the laser sensor information to create a local Costmap in front of the vehicle. The Costmap values represent the proximity of the vehicle to an obstacle using the geometry of the vehicle. The resulting map has the obstacles inflated by a security perimeter where the vehicle should not allow entering. Additionally, this map is used for TEB local planner in order to modify the local path based on the distance from the vehicle to the obstacles.

(vi) *TEB local planner* needs a global plan, Costmap, and vehicle odometry. It publishes the local plan (at a specific rate) with a configurable lookahead distance, and additionally, it publishes the control commands *ackermann_cmd* with the type *ackermannDriveStamped.msg*. The selected lookahead distance is 10 m because it is suitable to avoid big obstacles like other vehicles or groups of students.

(vii) *Obstacle manager* is responsible for sending a warning message to the movement manager for possible threats like pedestrians or obstacles in front of the vehicle. The type of this message is custom and has a list of obstacles with the information of distance and area for each obstacle. This node also returns the distance to the closest obstacle to take into consideration for emergency brake.

(viii) *Task manager* is in charge of selecting the task for the autonomous vehicle. For other projects, it is possible to configure different behaviors like platooning mode, drive between boundaries, or manual mode. This node receives information of the possible threats in the environment in order to stop the vehicle if necessary. This package publishes the reference velocity and reference steering angle.

(ix) *PID velocity* is responsible for creating the PWM necessary to the rear wheels DC motor. It takes as input the current velocity. The type of message is standard as *std_msgs/Float64.msg*.

(x) *Movement manager* is a driver in charge of checking all limits and possible wrong velocity messages. It sends the desired acceleration to the DC motor and the selected angle for the steering angle.

Transformation Trees (tf). One of the most important aspects to consider when working with vehicle odometry is the frame of reference and the coordinates. This node is in charge of transforming the movement of the vehicle from its ego-motion local reference to the global reference adding the initial position and orientation. Therefore, global and local path need the transformation between local coordinates and global coordinates. The *odometry manager* module is in charge of fusing the odometries and publishing the transformation between global and local coordinates using the initial localization in the map of the vehicle. Figure 7 shows all the frames involved in the architecture for one vehicle. The

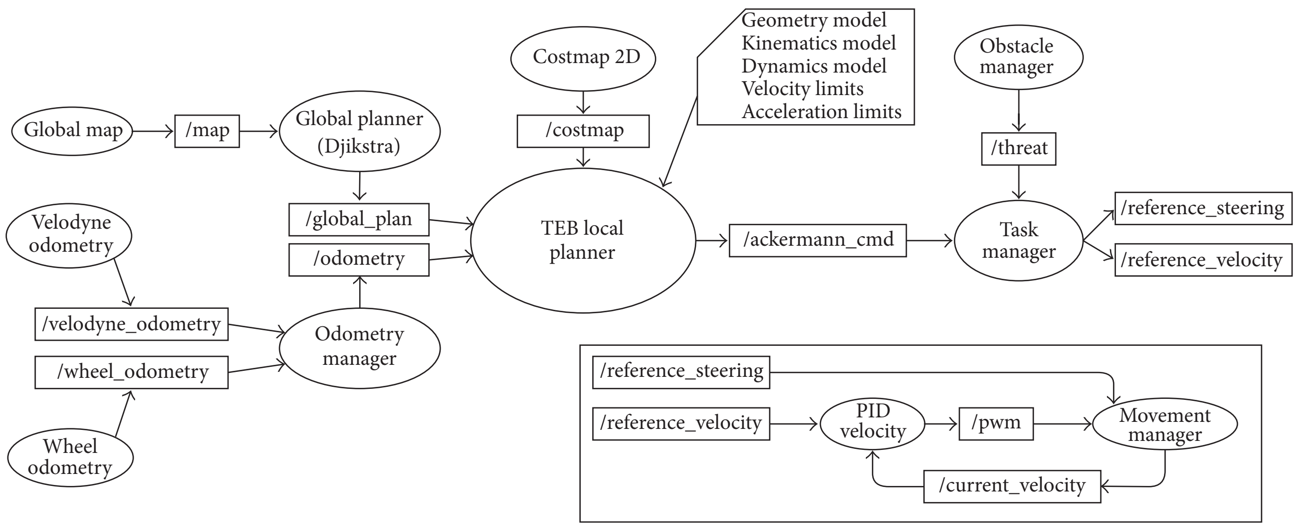


Figure 6: Software nodes sharing messages by topics in the partial software architecture.

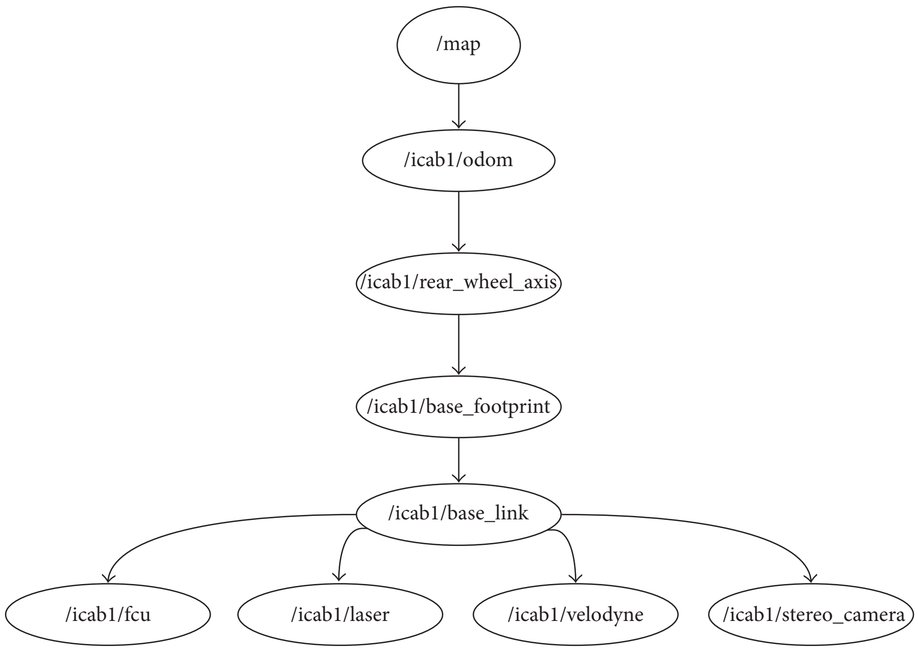


Figure 7: Transformation Tree for iCab1 showing all the sensors.

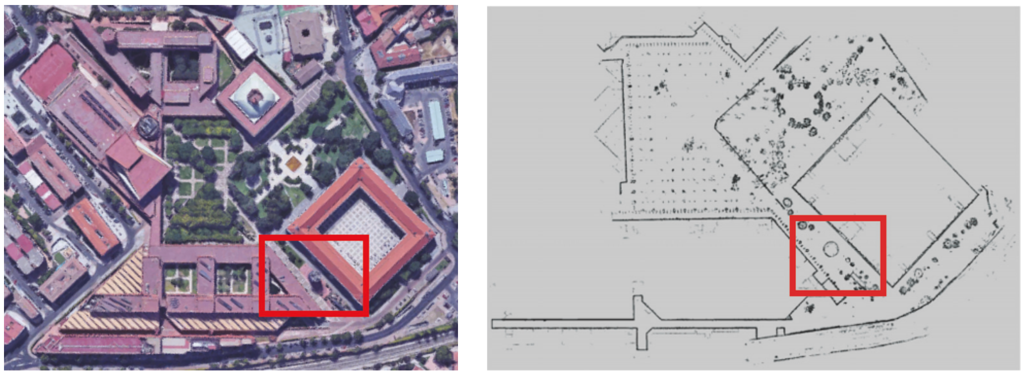
appropriate configuration of these transformations allows the system to collaborate with other coexisting heterogeneous vehicles under a vehicle namespace such as in this case icab1.

4. Experimental Setup

Aiming to test the path planner in the real platform, a simple test has been prepared where the vehicle navigates from a starting point and finishes in a goal point. The selected location takes place in a zone where a circular building is occluding the direct path trajectory. The data is recorded into a bag file, while the vehicle performs the experiment for posterior offline analysis. In Figure 8, the global map used to calculate the global plan is shown and the experiment zone is located inside of the red square.

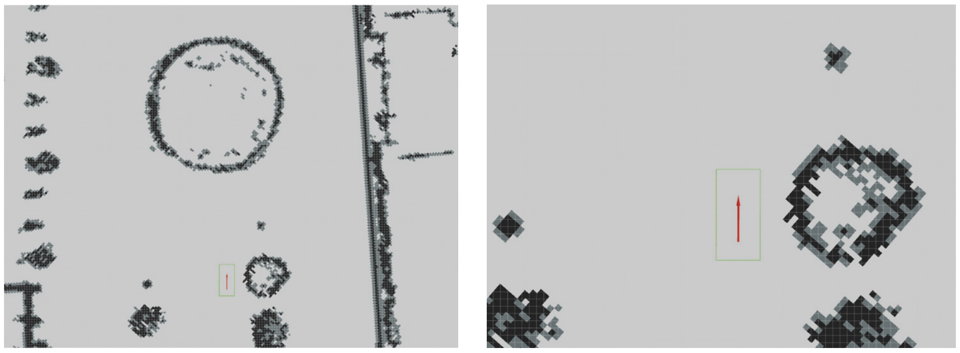
This map has been generated using the Lidar 3D points with the Lidar odometry and mapping [25] adapted to work with the correct axes reference. Afterwards, the point cloud has been processed to extract the floor at a specific height. The last step is to project this point cloud to the 2D space to have all the static obstacles (notice that the cup of the trees is taken into account as an obstacle due to the possible collision with the leaves and the top of the vehicle).

The representation of the vehicle, the maps, and the movements are easily analyzed using the RVIZ tool. Figure 9 describes the localization of the vehicle in the environment



(a) (b)

Figure 8: Satellite image of the Campus granted by Google Maps on (a). Campus gridmap on (b), the experiment is located inside of the red square.



(a) (b)

Figure 9: On (a), starting point of the vehicle oriented to the local odometry. On (b), zoom in the details of the vehicle geometry.

where the green rectangular shape represents the geometry and the red arrow represents the orientation.

After introducing the goal point, the representation of the lines for the paths is shown in Figure 10 where the blue line is the global path generated from the global planning using the current localization and the goal point. The input goal point is generated from the RVIZ visualizer using *move_base_msgs/MoveBaseGoal.msg* and clicking on the screen. The orange line is the local path generated from the TEB local planner and each cycle of the node is recalculated.

5. Results

All trajectories generated from the TEB local planner are evaluated offline by the data recorded, while the experiment is running.

5.1. Evaluation Metrics. The evaluation metrics used in this section are the Euclidean distance (DGL) between the Global Plan Waypoints (GPW) and Local Plan Waypoints (LPW) as shown in (1). The algorithm to find the best local plan is constantly updating it in order to avoid obstacles in the route or because the vehicle is not able to follow the previous local path.

$$\mathrm{DGL}_i = \sqrt{\left(\mathrm{GPW}_i - \mathrm{LPW}_i\right)^2} \quad \forall i \in Q. \tag{1}$$

5.2. Data Analysis. The experiment has been done by the lookahead distance configured to 10 m for the TEB algorithm because if an obstacle is above the global plan (a dynamic obstacle or a static obstacle which are not on the global map), this lookahead distance should be big enough to find another path even if it is far away from the global plan. Figure 11 represents the global plan generated with Dijkstra method on blue. In Figure 12, each line represents the local plan generated from the position of the vehicle while moving. The representation of multiple lines is due to the recalculation of the local trajectories. It is possible to appreciate the details of some lines that do not end into the goal point because of the lookahead distance configuration. Figure 13 represents the position of the vehicle at each moment in red dots. It is possible to appreciate all the graphs combined in the Figure 14 and with the details of the first three meters of the movement in Figure 15. Each red dot is the current odometry of the vehicle, the big blue line is the Dijkstra global plan, and each

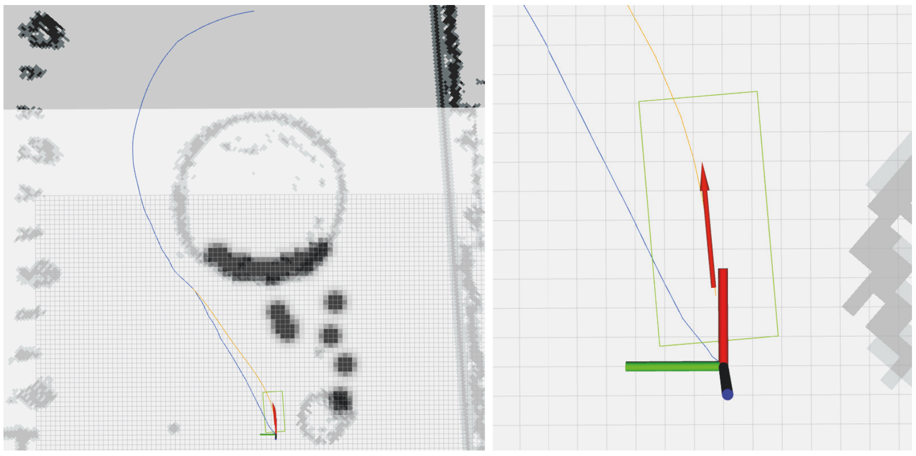


FIGURE 10: Global plan in blue. Local plan in orange. Overlay of global map and local Costmap 2D.

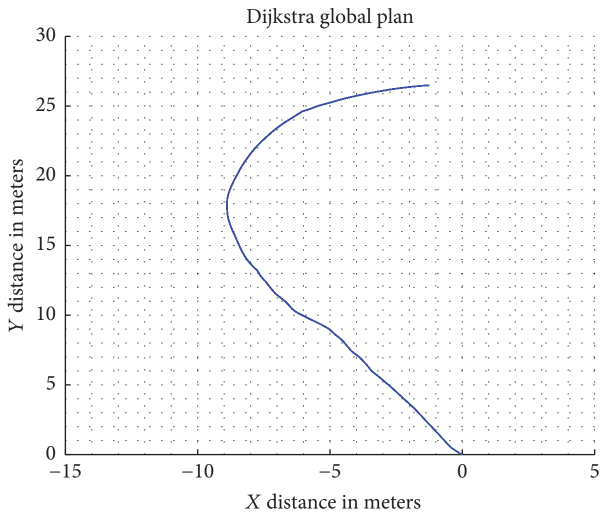


FIGURE 11: Global path.

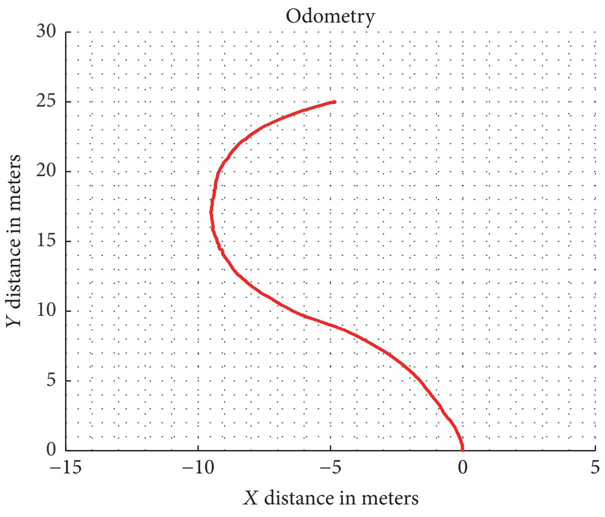


FIGURE 13: Odometry points.

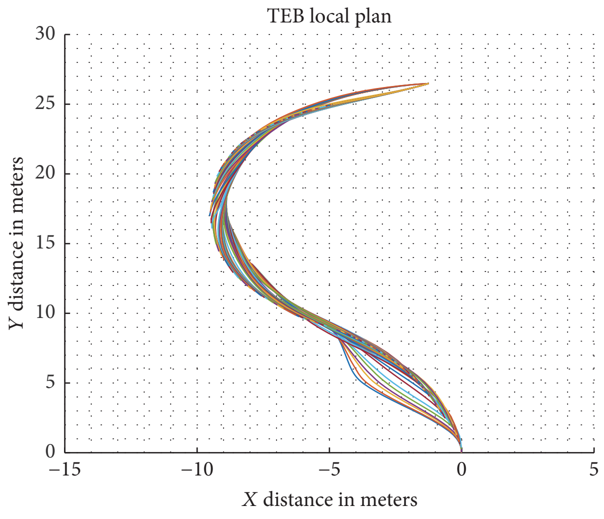


FIGURE 12: Local path.

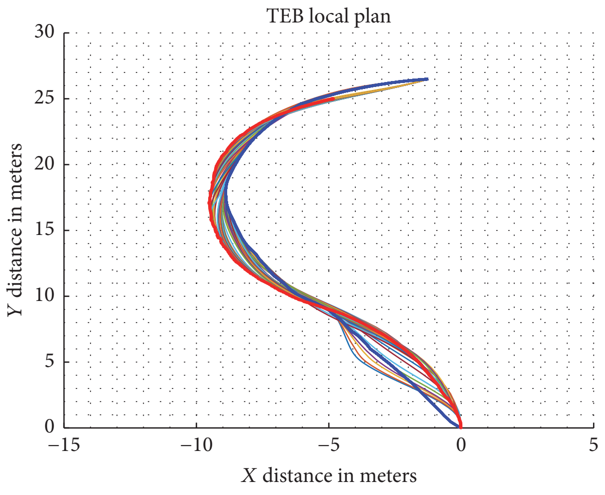


FIGURE 14: Combined.

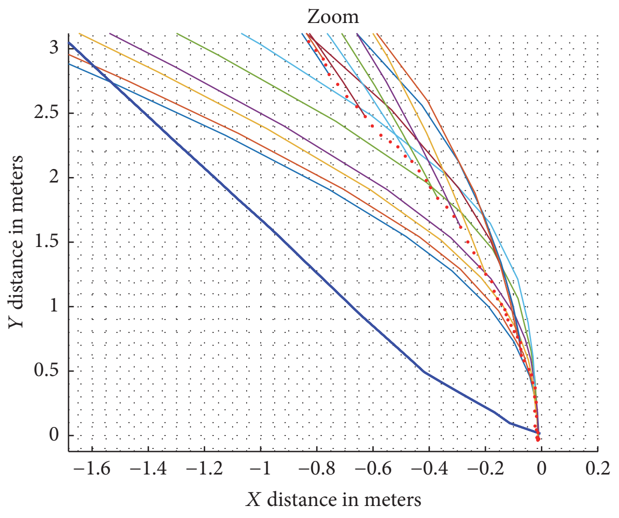


Figure 15: Combined zoom.

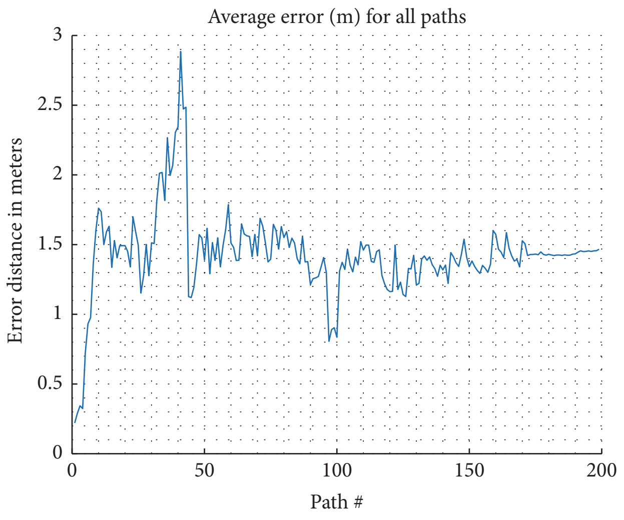


Figure 17: Average error.

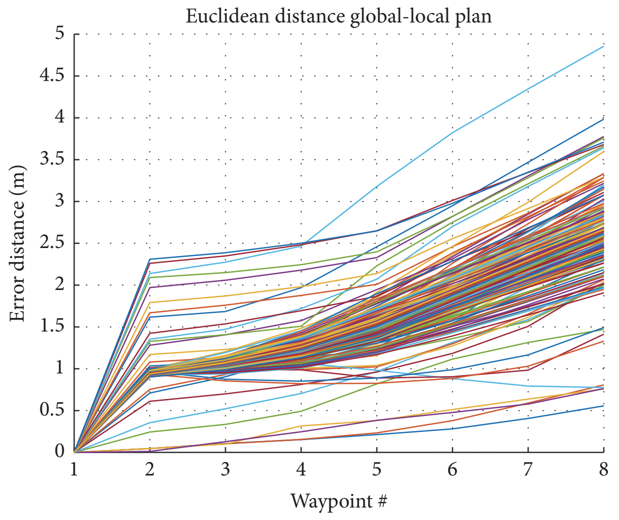


Figure 16: Error local plan, global plan.

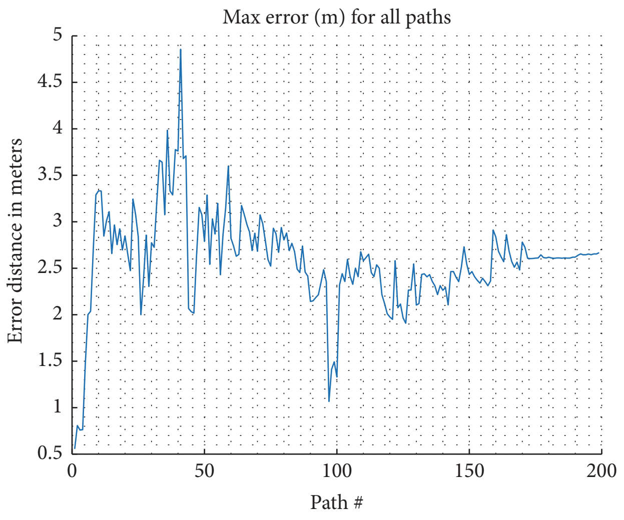


Figure 18: Max error.

thin line in multicolor corresponds to each TEB local plan. It is possible to appreciate the recalculations of the local plan in some dots where the vehicle is not able to follow the proposed trajectory (multicolor lines). Even if the vehicle does not follow perfectly the path due to the mathematical model variations configured into the TEB local planner parameters such as steering angle velocity, it seems that the header of the vehicle is different than in the calculated plan which follows into a wrong steering angle velocity configuration.

Euclidean Distance between Local and Global Path. The error between the global path and the local path determines how accurate the recalculation of the trajectory is. In Figure 16, each line represents the distance error for all local trajectories planned, while the vehicle was moving.

In Figure 16, the Euclidean distance between the first points of the local trajectory and the first points of the global trajectory from the position of the vehicle is described for all paths generated. The reason for this first point is because, after the recalculation of the new local path, the old path is not useful anymore. For the analysis, the seven first points have been selected for each local plan. Regarding this graph, it is possible to notice the incremental growth of the Euclidean distance point by point. The reason for this growth is because the vehicle is not able to follow properly the generated local path. Figure 17 is the result of computing the mean of the seven first values for every path. After the initial rise, there is a tendency around 1.42 m. The maximum Euclidean distance error for all paths generated between local path and global path is detailed in Figure 18. Due to the avoidance of an obstacle at the beginning, the local path deviates further from the global path.

Speed and Steering Angle. For the analysis of the control commands of the speed and steering angle granted by the TEB local planner module, Figure 19 shows the values in

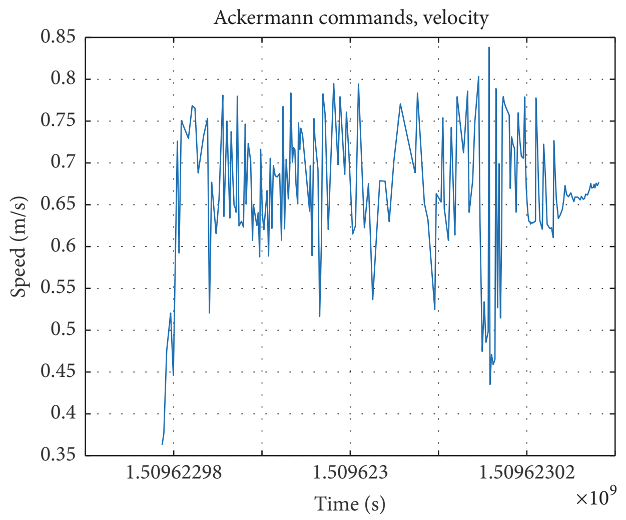


FIGURE 19: Speed.

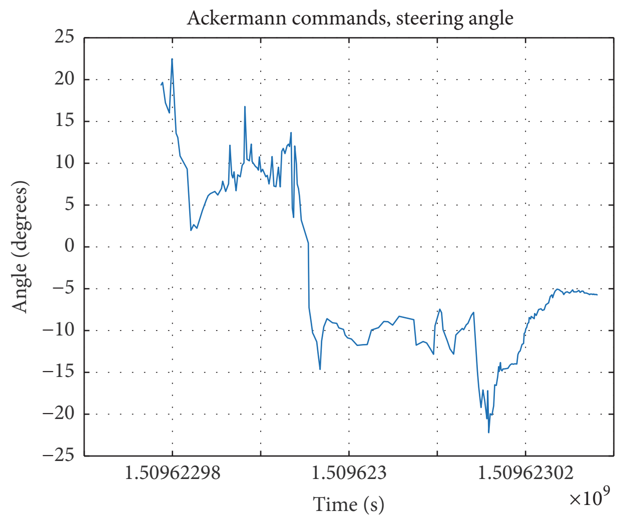


FIGURE 20: Steering.

real time provided to the vehicle. For the steering angle in Figure 20, it is possible to appreciate the variations of the commands each time the local plan is recalculated. At the beginning of the experiment, the steering angle provided is positive (turn to the left) until the vehicle reaches the position in the world where the circular building is left behind and starts to turn to the right with a negative angle.

6. Conclusions and Future Work

The first conclusion of this work determines that the use of Time Elastic Bands is a good choice even if the model for the kinematics of the vehicle is not precisely adjusted or when the vehicle is out of the path. This algorithm is useful in both of these events and when a dynamic obstacle is in front of the vehicle because of the continuous update of the local path. The second conclusion is that this method has been tested in a real vehicle where the proper configuration of the kinematics, dynamics, and geometry model could differ from the real model. This issue generates trajectories slightly different because the expected movement of the vehicle and the real movement of the vehicle are lightly different. Moreover, the TEB local planner generates the commands of velocities and steering angles that allow the vehicle to reach the goal.

For future work, more experiments are planned such as the comparison between this method and the Dynamic Window Approach and changing the control over the line generated with different path followers such as Pure Pursuit, Optimal Path Controller based on LQR, or Follow the Carrot. More complex scenarios will be added where pedestrians intersect with the trajectory generated and the vehicle should act accordingly, avoiding the pedestrian over the correct side in order to test the aforementioned solution for the elastic bands where a pedestrian is crossing in front of the vehicle.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is supported by Madrid Community Project SEGVAUTO-TRIES (S2013-MIT-2713) and by the Spanish Government CICYT Projects (TRA2013-48314-C3-1-R, TRA2015-63708-R, and TRA2016-78886-C3-1-R).

Supplementary Materials

The supplementary material of this work consists of a video of the real performance of the vehicle in the place where the experiment was done. In the video, it is possible to appreciate the visualizer RVIZ from ROS with two more windows. The top right window displays the left camera of the stereo vision system mounted on the vehicle. Below this window, the laser readings converted into a local map is placed. The background is composed of the global Costmap 2D and the overlay of the local Costmap 2D. On this map, the black squares correspond to obstacles and the color lines describe the trajectories. The blue line represents the global path generated by the Dijkstra algorithm and the orange represents the local path generated by the Time Elastic Bands algorithm. The green square on the bottom of the video represents the dimensions of the vehicle. As soon as the vehicle moves, the red arrows represent the movement and orientation of the vehicle and all of them are superimposed on each other because of the update of the odometry. The vehicle is able to navigate from one point to another without any collision. *(Supplementary Materials)*

References

[1] A. Broggi, M. Buzzoni, S. Debattisti et al., "Extensive tests of autonomous driving technologies," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1403–1415, 2013.

[2] J. Ziegler, P. Bender, M. Schreiber et al., “Making bertha drive-an autonomous journey on a historic route,” *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[3] B. Chen and G. Quan, “NP-hard problems of learning from examples,” in *Proceedings of the 5th International Conference on Fuzzy Systems and Knowledge Discovery*, pp. 182–186, IEEE, October 2008.

[4] M. Khristamto, A. Praptijanto, and S. Kaleg, “Measuring geometric and kinematic properties to design steering axis to angle turn of the electric golf car,” *Energy Procedia*, vol. 68, pp. 463–470, 2015.

[5] J. Canny, “A new algebraic method for robot motion planning and real geometry,” in *28th Annual Symposium on Foundations of Computer Science*, pp. 39–48, IEEE, 1987.

[6] P. Bhattacharya and M. L. Gavrilova, “Voronoi diagram in optimal path planning,” in *Proceedings of the 4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD ’07)*, pp. 38–47, July 2007.

[7] S. Skiena, “Dijkstra’s algorithm,” in *Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, pp. 225–227, Addison-Wesley, Boston, Mass, USA, 1990.

[8] R. Dechter and J. Pearl, “Generalized best-first search strategies and the optimality of A^*,” *Journal of the ACM (JACM)*, vol. 32, no. 3, pp. 505–536, 1985.

[9] D. Gelperin, “On the optimality of A^*,” *Artificial Intelligence*, vol. 8, no. 1, pp. 69–76, 1977.

[10] M. Šeda, “Roadmap methods vs. cell decomposition in robot motion planning,” in *Proceedings of the 6th WSEAS International Conference on Signal Processing, Robotics and Automation*, pp. 127–132, World Scientific and Engineering Academy and Society (WSEAS), 2007.

[11] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, “Evolutionary artificial potential fields and their application in real time robot path planning,” in *Proceedings of the Congress on Evolutionary Computation (CEC 00)*, vol. 1, pp. 256–263, IEEE, July 2000.

[12] S. M. LaValle, *Rapidly-exploring random trees: A new tool for path planning*, 1998.

[13] S. X. Yang and C. Luo, “A neural network approach to complete coverage path planning,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 1, pp. 718–725, 2004.

[14] M. Ben-Ari and F. Mondada, “Mapping-based navigation,” in *Elements of Robotics*, pp. 165–178, Springer, 2018.

[15] C. Rosmann, W. Feiten, T. Wosch, F. Hoffmann, and T. Bertram, “Efficient trajectory optimization using a sparse model,” in *Proceedings of the 2013 6th European Conference on Mobile Robots, ECMR 2013*, pp. 138–143, IEEE, September 2013.

[16] S. Gim, L. Adouane, S. Lee, and J.-P. Dérutin, “Clothoids composition method for smooth path generation of car-like vehicle navigation,” *Journal of Intelligent & Robotic Systems*, pp. 1–18, 2017.

[17] J. P. Rastelli, R. Lattarulo, and F. Nashashibi, “Dynamic trajectory generation using continuous-curvature algorithms for door to door assistance vehicles,” in *Proceedings of the 25th IEEE Intelligent Vehicles Symposium, IV 2014*, pp. 510–515, IEEE, June 2014.

[18] J. A. Reeds and L. A. Shepp, “Optimal paths for a car that goes both forwards and backwards,” *Pacific Journal of Mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[19] A. Piazzi, C. G. Lo Bianco, M. Bertozzi, A. Fascioli, and A. Broggi, “Quintic G_2-splines for the iterative steering of vision-based autonomous vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 27–36, 2002.

[20] C. Rösmann, W. Feiten, T. Wosch, F. Hoffmann, and T. Bertram, “Trajectory modification considering dynamic constraints of autonomous robots,” in *Proceedings of the 7th German Conference on Robotics; ROBOTIK 2012*, pp. 1–6, VDE, 2012.

[21] P. Marin-Plaza, J. Beltran, A. Hussein et al., “Stereo vision-based local occupancy grid map for autonomous navigation in ros,” in *Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP)*, vol. 3, pp. 703–708, 2016.

[22] R. Moliner and R. Tanda, “Tool for robust tuning of PI/PID controllers with two degree of freedom,” *RIAI: Revista Iberoamericana de Automatica e Informatica Industrial*, vol. 13, no. 1, pp. 22–31, 2016.

[23] R. Arnanz, F. J. García, and L. J. Miguel, “Methods for induction motor control: State of art,” *RIAI - Revista Iberoamericana de Automatica e Informatica Industrial*, vol. 13, no. 4, pp. 381–392, 2016.

[24] A. Kokuti, A. Hussein, P. Marin-Plaza, A. De La Escalera, and F. Garcia, “V2X communications architecture for off-road autonomous vehicles,” in *Proceedings of the 2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pp. 69–74, IEEE, Vienna, Austria, June 2017.

[25] J. Zhang and S. Singh, “LOAM: Lidar Odometry and Mapping in Real-time,” in *Robotics: Science and Systems*, vol. 2, 2014.