Autonomous Driving of Competition Robot - Localization and Environment Perception

Rúben Daniel Nunes Capitão
*Universidade de Lisboa, Av. Prof. Dr. Cavaco Silva 13, 2744-016 Porto Salvo, Portugal
Email: rubencapitao@gmail.com

***Abstract*—With the objective of competing in the Autonomous Driving challenge of the Portuguese Robotics Open 2018, an autonomous mobile robot based on a remote-controlled car was entirely designed, implemented and tested, including its hardware and software. Given its broad scope, the full extent of the competition challenges is covered across two parallel theses. The present thesis focuses, on the hardware part, on the electronics of the robot and the localization and perception of the environment software modules. It proposes a different approach from traditional lane-following techniques. In the thesis approach, the robot is fully aware of its surroundings and about its absolute localization. Although the track is planar, i.e., without any surrounding walls, a novel algorithm is developed to generate virtual walls over the lines of the track enabling occupancy-grid based localization using Monte Carlo Localization. The same algorithm is capable of identifying the track and segment it from the present obstacles standing on it, allowing their detection. To conclude, the recognition of semaphores and traffic signs based on state-of-the-art approaches, namely, image processing and machine learning, are also developed.**

***Index Terms*—Autonomous Driving; Localization; Perception of the Environment; Robotics Operating System.**

I. Introduction

Due to the increasing number of deaths and injuries caused by human errors in the public roads, the demand for autonomous driving vehicles has started to emerge. Since those aren't fully trusted by all citizens yet, it's been made an increasing effort to demonstrate the applicability and advantages of self-driving vehicles on the public roads.

At a national level, the Autonomous Driving challenge of the Portuguese Robotics Open has been promoting the development of the literature in this field and raising the awareness of the audience to this area.

The objective of the competition is to implement a fully autonomous robot capable of accomplishing the basic tasks of navigation on an environment very similar to the reality with intersections, zebra-crossings, semaphores, traffic signs, obstacles, tunnels, under-construction zones, and parking areas.

Given the board spectrum of the challenges, the problem was split into two categories: Localization and Environment Perception and Navigation and Decision-Making. The first is covered on the present document, being the second developed on a parallel work.

This document focuses on the development of the electronics and necessary sensors to deploy an autonomous driving research platform for the current and subsequent years of the competition and the software modules required to carry the localization and environment perception tasks.

II. Literature Review

A. Robot Pose Estimation

1) Robot Localization: To address the problem of estimating the robot's pose relative to the surrounding environment, the probabilistic Bayesian Framework [1] is utilized across virtually all literature, where the computation of the robot's probability density function is performed in two steps: the prediction and correction. The prediction step takes into consideration the command provided to the robot and the robot's kinematic model to estimate the position of the robot. The correction step incorporates the sensor's readings to overall certainty of the robot's position.

As mentioned, several approaches have been successfully applied to decrease the overall complexity of the Bayesian Framework original algorithm. One of the most used methods is the Monte Carlo Localization [2] since it's multi-modal, i.e., it allows global localization of the robot within a map and addresses the problem of the "kidnapped" robot due to the use of Particle Filters.

2) Sensors used for robot localization and Sensor fusion: The most used sensors for pose estimation in the literature are wheeled odometry, Inertial Measurement Units (IMU), LIDARS, GPS and cameras. However, due to the monetary resources and challenges of the environment, only wheeled odometry, IMUs and cameras can be used.

Both sensors have their advantages and downfalls, thus, the strengths of each are typically combined using sensor fusion algorithms. The most used sensor fusion algorithm is the Kalman Filter [3] due to its small computational requirement and ability to mix data at different rates.

B. Perception of the Environment

1) Environment representation: Different types of representation of the environment are described in the literature. The most commonly used with Monte Carlo Localization is the Occupancy-Grid. As the name implies, the map is subdivided in a grid with the value of the occupancy. Each cell can be either free, occupied, or unknown. The problem is that these algorithms are only applicable when there's an available laser scan output. The present work addresses this problem further ahead.

2) Detection of obstacles: For the detection of the obstacles, LIDAR sensors are excluded due to its cost. The most common way to deal with this problem aside from the mentioned sensor is to either estimate the depth using a

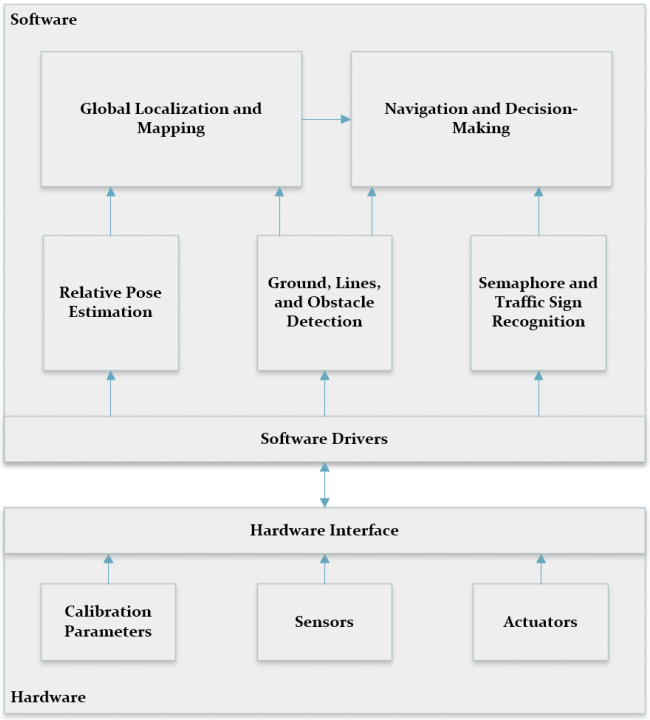


Fig. 1. Proposal of functional architecture, including both Hardware and Software modules.

monocular camera or using a stereo or RGB-Depth sensor, like the Kinect. The latter has demonstrated good practical results since the depth calculation is not necessary, decreasing the computational effort while also providing a good accuracy [4].

3) Traffic sign detection and recognition: A high number of approaches to address this problem have been proposed, and these are typically segmented in two distinct problems: the detection and the recognition. The detection of a traffic sign is usually performed on a cluttered environment where both color and shape based techniques are applied or a combination of both [5]. For the detected shapes, colors, or both, the recognition of the interior part of the hypothetical sign is carried mainly with three approaches: image processing techniques, Support Vector Machines, or Neural Networks. Machine learning approaches show results above 99 % and even reaching 100 % for real data sets [6].

III. System Architecture

A high-level block diagram depicting the covered modules and their connections is shown in Fig. III.

A. Hardware

All of the hardware is supported by a 1:8 Radio Controlled car, defined by a non-holonomic model to mimic as much as possible a real-world scenario. The robot is depicted in the Fig. III-A.

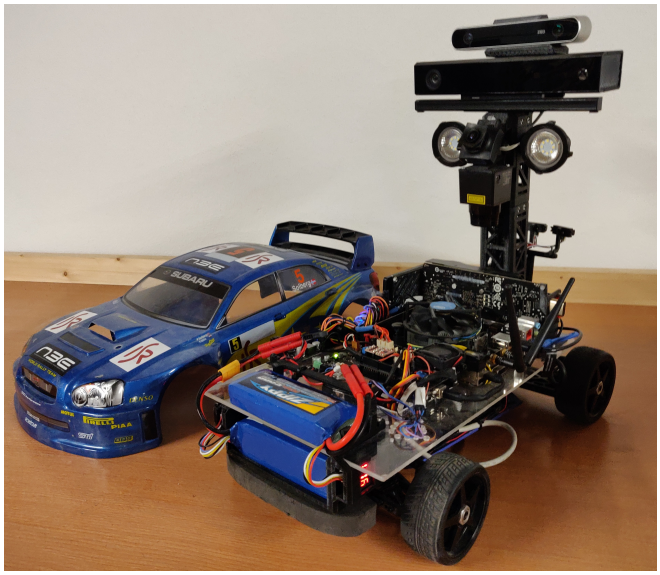


Fig. 2. Robot's chassis and hardware overview.

1) Actuators: The actuators included on the car-like robot to act like a real vehicle and carry all challenges are an electric motor, one servo for the direction, another servo for the brake, and lights.

2) Sensors: The sensors included in the robot are: a Kinect One, which combines color images with depth information that is used to generate the virtual walls by combining the two different sources of information and to detect obstacles; a wide-angle camera that is able to keep track of the semaphores and traffic signs present in the environment without having to be directly facing those; a ZED stereo cameras that is used as the main source of odometry since the mechanical characteristics of the used car-like frame render the application of wheeled of motor odometry unfeasible; an MPU-9250 which includes an accelerometer, gyroscope, and compass to increase the orientation accuracy of the pose estimate; and, finally, infrared sensors, which allow safe maneuvers in reverse gear.

3) Hardware interface: All of the sensors and actuators are connected to the hardware interface, with the exception of the camera, which can be directly connected to the computing platform. The hardware interface is responsible for making the bridge between the low-level sensors and actuators.

4) Computing Platform: A mini-ITX format motherboard is included in the frame to fit inside the robot frame. A dedicated computing platform is designed for the competition to include only the necessary aspects, namely: computing performance. With this in mind a powerful CPU and GPU are also found in the robot.

B. Software

1) Operating System: On top of the computing platform, the Linux-based Robotics Operating System (ROS) is installed. The framework's main points of attraction are its distributed and modular design, simulation capabilities, its open-

source community, and built-in visualizer, providing bootstrapping capabilities to both starters and experienced users. One or more ROS nodes represent each software module.

2) Sensor calibration parameters: To reduce the overall uncertainty associated with the sensor's measurements, a dedicated module to calibrate the sensors and discover its intrinsic characteristics is created. The calibration is to be performed before the competition. However, the calibration information is to be used during the execution of the assigned tasks to ensure maximum precision and accuracy of the sensors.

3) Ground, lines, and obstacles detection: This module is subdivided in its three constituents. The first segments the Kinect's point cloud by identifying the open areas ahead of the robot and the obstacles standing on it. The navigable area is then assessed by a second module which implements the novel algorithm and generates a virtual laser scan from the track's lines. The last one determines the obstacle point cloud and informs the Navigation and Decision-Making modules about the existence and location of obstacles.

4) Relative Pose Estimation: The relative pose estimation node combines the pose information of the different sources capable of providing a position or orientation estimate, providing a more accurate pose than each sensor alone.

5) Global Localization and Mapping: With the relative pose estimation and the virtual walls, a map is created by traveling on the track several times before the competition. Since the environment is static, this map is used during the competition for localization and navigation purposes.

6) Semaphore and Traffic Sign Recognition: By assessing the output image of the wide-angle camera, this module recognizes the presence of semaphores and traffic signs and informs the Navigation and Decision-Making modules of their presence.

IV. Implementation

A. Hardware Interface

A custom Printed Circuit Board (PCB) is designed support the smaller sensors and actuators, to provide an easy interface for all sensors and actuators, and to distribute the power to all electronic components inside the robot's chassis, with the exception of the of the cameras which already possess a standard Universal Serial Bus (USB) connection, and the computing platform and motor which have its dedicated power supply and driver.

As it can be observed in Fig. IV-A, the PCB has the same shape as the microcontroller, an Arduino Mega, so that it can be directly mounted over it, saving up space and in unnecessary cabled connections.

With all the necessary connections made and sensors powered, the microcontroller takes advantage of the *rosserial* [7] to send and receive data directly from ROS nodes. The Arduino is programmed to iteratively check the latest value of each sensor and output it to the computing platform for its assessment. When data is received from the computing platform, it stops the current execution and applies the latest command to the actuators.

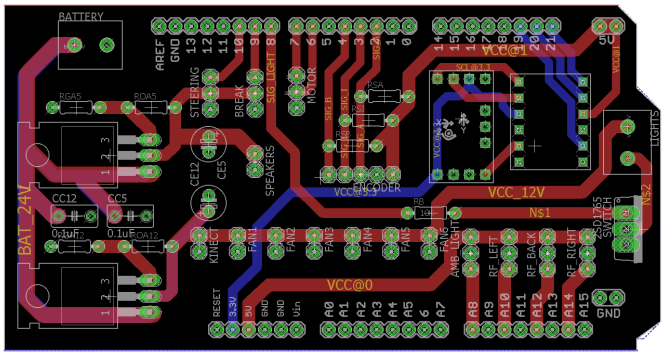


Fig. 3. Designed Printed Circuit Board.

B. Sensor Calibration

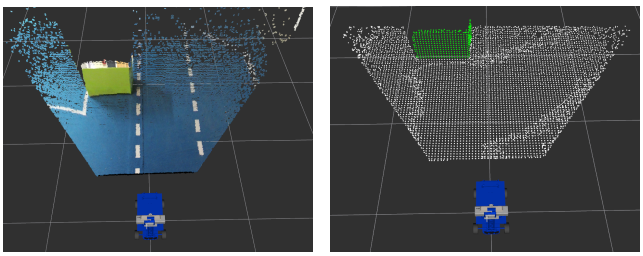
A specific Arduino program is developed to obtain, in a semi-automated way, the low-level sensor's intrinsics, namely: the IMU's bias and scale sensor factors and the variance of the infrared signal based on the distance and object's surface texture.

For the remaining sensors, only the Kinect One needs to be calibrated. The program to accomplish this it's already included in the used *iai_kinect2* [8] driver.

C. Ground, lines, and obstacle detection

The current implementation of this module, as described earlier, is subdivided into three parts.

The first module takes advantage of the Point Cloud Library to downsample the original Kinect's point cloud and estimate the existing planes using the RANdom SAmple Consensus (RANSAC) method. If the estimated plane is within a defined threshold of the xy plane, the plane is considered as navigable area or ground, being the remaining points considered obstacles. The inputs and outputs of the algorithm can be visualized in Fig. 4.



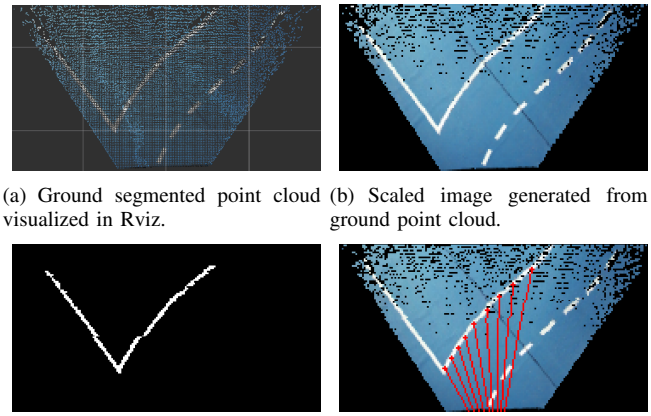
(a) Kinect's original point cloud.

(b) Segmented and downsampled ground point cloud, represented in white, and the obstacles point cloud, represented in green.

Fig. 4. Visual description and representation of the ground and obstacles segmentation algorithm.

With the ground point cloud, a scaled image is created and, using image filtering techniques, only the continuous white lines remain in the image. After that, virtual beams starting from the robot are drawn over the scaled image until a white pixel is identified. With this, a virtual laser scan is created,

generating the mentioned virtual walls based on the track's lines. The process can be visualized in the Fig. 5



(a) Ground segmented point cloud visualized in Rviz.

(b) Scaled image generated from ground point cloud.

(c) Color filter application and removal of the dashed line and noise.

(d) Visual representation of the virtual beams in red.

Fig. 5. Visual description and representation of the generation of the virtual laser scan from an input colored point cloud.

From the remaining points that are considered obstacles, another module assesses the height of the points and, if those are within a specified range, a usual laser scan is output to inform the Navigation and Decision-Making modules about the position of the obstacle. This process is depicted in Fig. 6.

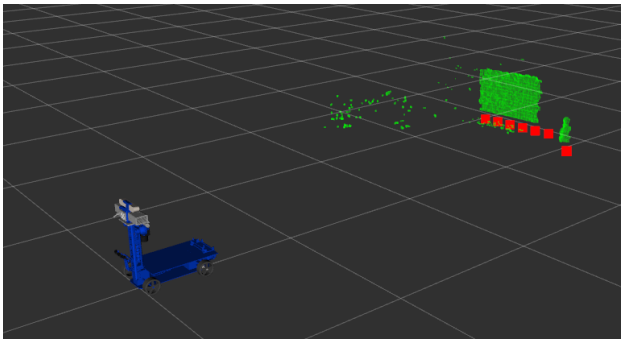


Fig. 6. Visual demonstration of the output of the obstacle detection node, with red color, from the input obstacles point cloud, with greed color.

D. Robot Pose Estimation

Since the environment is planar, i.e., the subtle nuances of the floor can be ignored, in order to produce an accurate 2D position and orientation estimation, the IMU orientation values are combined with the visual odometry position and orientation estimates of the ZED using the Extended Kalman Filter implementation of the *robot_localization* [9] package.

E. Global Localization and Mapping

In order to generate an environment map, *OpenSlam's GMapping* [10] algorithm is implemented. It solves the simultaneous and location problem using the Rao-Blackwellized PF. The key idea of this PF, is that both the trajectory and map are estimated based on the detected environment features, named observations, and the odometry measurements in two distinct steps. The first assesses the possible trajectories, while the second incorporates those hypothetical trajectories to assess the weight of the generated map particle.

By traveling the map several times, a map is automatically created and stored to be used during the competition challenges.

To assess the absolute localization within the generated map, the Adaptative Monte Carlo Localization of Fox et al. [11] is implemented through the usage of the *amcl* [12] package.

It's verified that both algorithms assume that the measurements of the laser scan are orders of magnitude more accurate than the odometry sensors. Since, in this case, the laser scan is virtually generated and given that the odometry is quite accurate, this assumption is not valid. Thus, to obtain optimal results, the expected odometry error had to be largely decreased.

F. Semaphores and Traffic Sign Recognition

Given that the semaphores' shapes are fairly simple and the number of shapes is small when compared with the available traffic signs, two different approaches are implemented.

For the semaphore recognition, an image processing method based on the state of the art approaches is implemented. It takes the input image, applies a color filter, followed by a template matching technique to assess the presence of a semaphore. This simple and pragmatic approach is demonstrated to be a success when the data set is small, and the localization and control of the robot are both stable. However, this cannot be assumed for all occasions.

For the traffic sign recognition, for the reasons shared above, a machine learning method is used instead. You Only Look Once (YOLO) neural network of Redmon and Divvala [13] is chosen due to its method of operation.

Contrary to other neural networks approaches, where a region based classifier is used to search for an object within an image several times, the authors of YOLO solve this problem in a single pass over the image. This is accomplished by segmenting the image in a grid where, for each cell, a bounding box for a hypothetical object is created and a class probability is estimated. By multiplying the probability of the class with the confidence of the bounding box having an object inside it, a weight is calculated to each bounding box. The weights are thresholded and, when above a specified value, the bounding box and identified class are considered as detected.

The current implementation of the neural network divides the problems into two layers, the detection of the class of the traffic sign (mandatory, informative, danger) and, from the output bounding box of the first layer, the interior part of the detected sign is assessed.

V. Evaluation

Each node's performance is evaluated along with the overall performance of the system in the competition environment. The main characteristics on which each module can be evaluated are:

- Accuracy of the obtained values.
- Computation cost and real-time applicability.
- Ratio between correct and incorrect operations.

A. Ground segmentation node

For a data set of 10000 point clouds, the algorithm that it's always able to detect the ground, either when stopped or moving, in a single iteration. It's also demonstrated to work at around 12.4 Hz, which provides a rate higher than the most laser scan sensors, proving its real-time applicability.

B. Road lines detection and generation of Virtual walls

From the performed tests, it's demonstrated only to be applicable if a line is within approximately 2.5 m from the car. At that distance, it's able to detect a line with an expected error of $\pm$ 0.01985 m for 68 % of the time.

C. Obstacle Detection

Although the Kinect's field of view is fairly limited, it compensates that with great precision of detected obstacle. During the captured data set of 25000 points, ranging from 1 m to 5 m, the expected error of the Kinect, for the worst case scenario, is very close to 1 cm for 99.7 % of the time, which is considerably accurate considering its price tag.

D. Semaphore Detection

The pragmatic implementation of the template matching techniques for the detection of the semaphores, demonstrate to have, in the best case scenario, a success rate of 97.8 %. However, if the image is slightly rotated, the template matching technique's success starts to deteriorate, reaching the lowest value of 77.4 % success rate for the Parking semaphore.

E. Traffic sign recognition

Contrary to the semaphore detection technique, the implementation of the two neural network layer demonstrates the best possible results, where it is always able to detect the existing traffic signs with 100 %, even when those are rotated, or turned upside down, which would not happen on the image processing approach.

F. Global Map

After creating 10 maps with the global mapping node, the best map is selected and presented in Fig. 7.

The figure is very similar to the real environment, providing satisfactory results. Even though there are some artifacts and errors, it provides a scalable solution to the problem, should the track change in the subsequent years.

G. Absolute Localization

From the test carried, it's determined that the absolute localization error increases exponentially with the increase of the velocity. Due to the ambitious accuracy defined by the specifications, the robot is only able to meet the position accuracy at 0.5 m/s, although it's able to meet the desired orientation accuracy at 1.5 m/s.

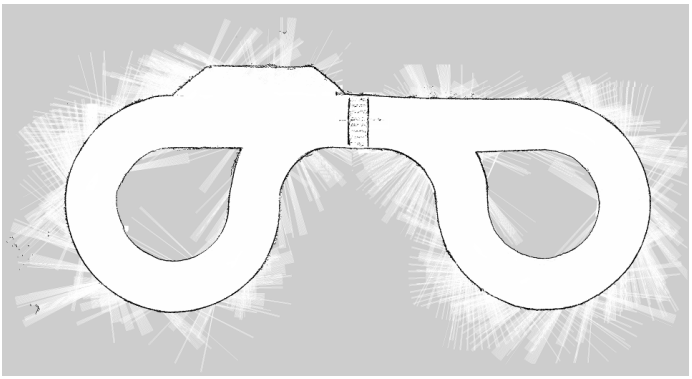


Fig. 7. Global map of the track captured by GMapping on the test environment.

Even though the specifications are not met for higher velocities, the tests performed demonstrate that the error of the relative pose estimation node, after a full run, which are two full laps to the track, sums up to nearly 1 m of drift, going out of the boundaries of the track after a completing half of a lap. These results can be observed in the Fig. 8 and also demonstrates the success of the absolute localization node.

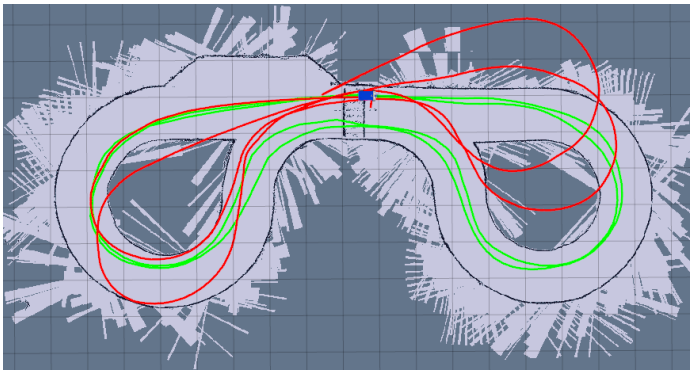


Fig. 8. Visual demonstration of path provided by the global localization, represented with the green color, and relative pose estimation, represented with the red color.

H. Performance of the system in the competition environment

In its first iteration at the Robotics Open 2018, help in April, the robot was able to complete the majority of the challenges, achieving the 2nd place, very close to the 1st position.

This proof-of-concept validated the success of the current approach. Contrary to the other teams' approaches, where the robot typically follows a line until a specific condition is met the present system is fully aware of its absolute localization within the track along with the absolute position of the obstacles within the track, allowing it to plan the best trajectory ahead of time instead of reacting to the presence of the mentioned specific conditions of obstacles.

During the FNR competition, the car's target speed was set to 1 m/s as the system was still under development. The nodes were not fully optimized and, under some circumstances, the

robot's absolute position would perform jumps of approximately half of a meter. This heavily affected the navigation and decision-making modules and, for that reason, the target speed was set to a lower value than specified. On the first iteration of the system, it did not include the traffic signs recognition node, thus, the mentioned challenge was not completed.

To conclude, as also demonstrated on the nodes' evaluation sections and in Francisco's thesis [14], the final iteration of the robot can successfully execute all the challenges within the defined specifications except the target speed. Even though it's able to navigate at that speed, given the length of the track, the localization error and control instability, the car-like robot commits a large number of penalties. Since these heavily affect the scores, lower speeds are targeted. Nevertheless, the speed of 1 m/s almost doubles all of the other team's speeds. This demonstrates that even though the specified goals are ambitious, the objectives have been met.

VI. Conclusions

A proposal of a system with the objective of competing in the Autonomous Driving challenge of the Portuguese Robotics Open is defined focusing on the necessary hardware and electronics that serve as basis to the designed autonomous driving robotics platform and on the necessary modules to provide an absolute localization within a known map and to perceive the surrounding objects of the environment, being those obstacles, semaphores, traffic signs, or construction cones.

To address this problem, the state of the art sensors and methods used in the literature are reviewed and implemented. The proposed system takes advantage of the state of the art methods to assess its absolute position within a map and to plan the optimal trajectory, presenting an uncommon and unique solution in the event's history.

All the necessary steps to create an autonomous driving vehicle are outlined in this thesis, and it's completed in [14]. It starts on the design of the basic electronic components and circuitry to power and support the low-levels sensors and actuators, goes through the creation of a hardware interface that bridges between the hardware and the Robotics Operating System, and finalizes with the implementation and evaluation of the proposed software nodes.

To address the problem of lack of features of the environment to apply a global localization algorithm, a novel technique to generate virtual walls based on the track's lines is developed, enabling Occupancy-Grid based localization algorithms like MCL. With these virtual walls, it's possible to generate virtually any map that is filled with continuous lines and localize the robot in it. This approach provides a scalable solution should the map of the environment change in the subsequent years. Additionally, using the same sensor, the Kinect One, it's also presented an approach to detect the navigable path and the obstacles standing in it and two distinct methods to recognize the different signs present on the environment, namely, the semaphores and the traffic signs.

The developed system can localize the robot, for a constant velocity of 1 m/s with an expected error of $\pm$ 0.0263 m in the position and $\pm$ 1.375 ^{o} in the orientation. The robot can meet the specified $\pm$ 0.025 m at lower velocities, however, during the challenges, the robot can finalize within the expected stopping area of 0.05 m most of the times. Given that the speed is a big factor on the score and given that the expected error at 1 m/s is very close to the specified limit, this is set as the target velocity. Even though the robot can reach the defined target speed of 2 m/s, this ambitious velocity is almost impracticable due to the size of the track, the localization error at this velocity, and the instability of the control at this pace.

Regarding the detection of obstacles, those are detected within a range of 0.5 m to 5 m with an expected error of approximately 0.01 for 99.73 % of the time, which is five times less than specified, and at a rate of 12 Hz, which is more than double of the specified rate.

The pragmatic solution presented for the detection of the semaphores is capable of fulfilling the objectives of the competition and is demonstrated to be a viable solution when the data set is small and when the control of the car is optimal, i.e., if the car is able to directly point to the semaphores.

The neural network approach for the traffic signs is capable of achieving a 100 % success rate and could also be used for the semaphores to increase the success rate when the semaphores are rotated, i.e., when the car is not directly pointing to the semaphores.

The first iteration of the designed system was showcased at Robotics Open competition, held in April 2018, achieving the 2nd place, very close in terms of score to the 1st position, confirming the success of the concept and its advantages. The final iteration of the autonomous driving system presents a more stable solution which is demonstrated to be able to address all of the competition challenges.

VII. Future work

Given that an academic robotics platform for autonomous driving is built, it can serve as a basis for a forthcoming work. Below, it's presented a list of the possible improvements on the presented work:

- Implementation of the *ground segmentation* and *virtual walls* nodes over GPU. Although the current implementation is able to achieve a rate of 12 Hz, it's only achieved by downsampling the Kinect's point cloud. It's possible that the accuracy of the algorithm increases if the *voxel size* of the downsampling process is reduced while maintaining a high output rate.
- Switching the implemented linear power regulators for switching regulators to decrease the overall energy dissipation of the circuit and, consequently, the temperature inside the robot frame.
- Validating if the inclusion of a wheel odometry system to each wheel, which was not possible due to the suspension of the used vehicle, increases the accuracy of the relative pose estimation.
- Evaluate the developed system in a different environment to assess the scalability of the developed solution.

REFERENCES

[1] H. W. Sorenson and A. R. Stubberud §, "Non-linear filtering by approximation of the a posteriori density," *International Journal of Control*, vol. 8, no. 1, pp. 33–51, 1968.

[2] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, vol. 2, pp. 1322–1328, IEEE, 1999.

[3] R. E. Kalman *et al.*, "A new approach to linear filtering and prediction problems," *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[4] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke, "Real-time plane segmentation using rgb-d cameras," in *Robot Soccer World Cup*, pp. 306–317, Springer, 2011.

[5] G. Piccioli, E. De Micheli, P. Parodi, and M. Campani, "Robust road sign detection and recognition from image sequences," in *Intelligent Vehicles' 94 Symposium, Proceedings of the*, pp. 278–283, IEEE, 1994.

[6] A. Shustanov and P. Yakimov, "Cnn design for real-time traffic sign recognition," *Procedia Engineering*, vol. 201, pp. 718–725, 2017.

[7]

[8] T. Wiedemeyer, "IAI Kinect2," 2014 – 2015. Accessed June 12, 2015.

[9] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*, Springer, July 2014.

[10] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[11] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," *AAAI/IAAI*, vol. 1999, no. 343-349, pp. 2–2, 1999.

[12]

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[14] F. Varandas, "Autonomous driving of competition robot - navigation and decision making," 2018.