Article

Evaluation and Testing Platform for Automotive LiDAR Sensors

Tiago Gomes [1,†]*, Ricardo Roriz [1,†], Luís Cunha [1,†], Andreas Ganal [2], Narciso Soares [2], Teresa Araújo [2], and João L. Monteiro [1,†]

[1] Centro ALGORITMI, Universidade do Minho, Portugal; {mr.gomes, ricardo.roriz, luis.cunha, joao.monteiro}@dei.uminho.pt
[2] Bosch Car Multimedia Portugal S.A., Automotive Electronics Division; {andreas.ganal, narciso.soares, teresa.araujo}@pt.bosch.com
* Correspondence: mr.gomes@dei.uminho.pt; Tel.: +351-253510180 (T.G.)
† Current address: Centro ALGORITMI, Escola de Engenharia - Universidade do Minho, 4800-058 Guimarães, Portugal

Abstract: The world is facing a great technological transformation towards full autonomous vehicles, where optimists predict that by 2030, autonomous vehicles will be sufficiently reliable, affordable and common to displace most human driving. To cope with these trends, reliable perception systems must enable vehicles to hear and see all the surroundings, being light detection and ranging (LiDAR) sensors a key instrument for recreating a 3D visualization of the world in real time. However, perception systems must rely in accurate measurements of the environment. Thus, sensors must be calibrated and benchmarked before being placed on the market or assembled in a car. This article presents an Evaluation and Testing Platform for Automotive LiDAR sensors with the main goal of testing not only commercially available sensors, but also sensor prototypes currently under development in Bosch Automotive Electronics division. The testing system can benchmark any LiDAR sensor under different conditions, recreating the expected driving environment to which such devices are normally subjected. To characterize and validate the sensor under test, the platform evaluates several parameters such as the field of view (FoV), angular resolution, sensor's range, etc. This project results from a partnership between the University of Minho and Bosch Car Multimedia Portugal, S.A.

Keywords: autonomous driving; LiDAR; perception systems; evaluation and testing.

1. Introduction

The world is undergoing an unprecedented technological transformation, where vehicles and autonomous driving systems are evolving at a breathtaking pace [1–4]. Optimistic predictions claim that by 2030, autonomous vehicles will be sufficiently reliable, affordable and common to displace most human driving, providing huge savings and benefits [5]. However, most of the vehicles in our roads today are still manually controlled, and to achieve full driving autonomy they must evolve through different levels of driving automation, as defined by the American Society of Automotive Engineers (SAE) [6]. While levels 0 - No Driving Automation, 1 - Driver Assistance, and 2 - Partial Driving Automation, still require the human driver to monitor the driving environment; with levels 3 - Conditional Automation, 4 - High Automation, and 5 - Full Automation, the automated system can autonomously monitor and navigate the driving environment.

Current level-2 vehicles are provided with advanced driver-assistance systems (ADAS), which can help the driver in several decisions upon situations that may compromise the safety of all occupants, assist in different parking tasks, provide traffic alerts, promote collision avoidance with other vehicles and objects, etc. Nonetheless, to cope

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2021 doi:10.20944/preprints202111.0562.v1

with these revolutionary trends, new solutions at the sensor level must be created to enable vehicles the ability to hear and see the surrounding environment. An autonomous vehicle requires reliable sensors to recreate an accurate mapping of the surroundings, which is only possible with multi-sensor perception systems relying on a combination of Radar, Cameras, and light detection and ranging (LiDAR) sensors [7–10], as illustrated in Figure 1. Radar sensors can provide (1) cross-traffic alerts, Blind Spot Assist features, and (2) Adaptive Cruise Control; the LiDAR sensor can be used to (3) translate the physical world into a 3D representation, achieving several distances with high levels of accuracy and precision; and Camera vision systems can help in features such as (4) object detection and classification, and (5) collision avoidance.

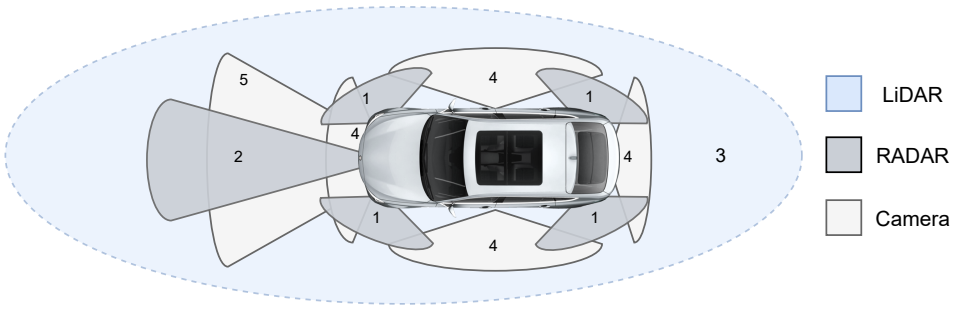


Figure 1. Perception system of a car.

LiDAR sensors are emerging as the state-of-the-art technology that must be mandatory on a perception system, since it enables a true 3D visualization of the surroundings through a point cloud representation in real time [11–13]. Accurate and precise measurements of the surroundings with a LiDAR can assist the perception systems in several tasks [9], e.g., obstacles, objects, and vehicles detection [14–16]; pedestrians recognition and tracking [17,18]; ground segmentation for road filtering [19]; among others [20]. The advances around LiDAR keep improving its measuring and imaging architectures [12,21,22]. Nonetheless, the measurements and the 3D point cloud of a LiDAR sensor can always be corrupted by several noise sources, e.g., internal components [23], mutual interference [24,25], reflectivity issues [26], light [11], adverse weather conditions [10,27,28], and others [29], making compulsory to test and analyze all sensor's characteristics before being placed on the market or assembled in a car.

This article presents an Evaluation and Testing Platform for Automotive LiDAR Sensors. The main goal of the testing platform is to test not only commercially available sensors, but also sensor prototypes that are under development in the Automotive Electronics division of Bosch Car Multimedia Portugal, S.A. The testing system is able to benchmark any LiDAR sensor under under real situations, created on a simulation/emulation environment, to recreate the expected driving conditions to which such devices are normally subjected. These conditions can be related to disturbances caused by different targets with different materials, compositions, reflectiveness, geometry, environmental and noise conditions, among others. In order to characterize and validate the sensor under test, the testing platform evaluates several parameters such as the field of view (FoV), angular resolution, sensor's range, etc. This article contributes to the state of the art with:

1) an evaluation and testing platform for testing several parameters of a LiDAR sensor for automotive applications;
2) a point cloud filter-based approach to evaluate several characteristics of a LiDAR sensor at the reception level;
3) a desktop and an embedded approach for deploying the testing platform software;
4) and the validation of the platform with the test and evaluation of a commercial off-the-shelf (COTS) LiDAR sensor.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2021 doi:10.20944/preprints202111.0562.v1

2. LiDAR Sensors for Automotive

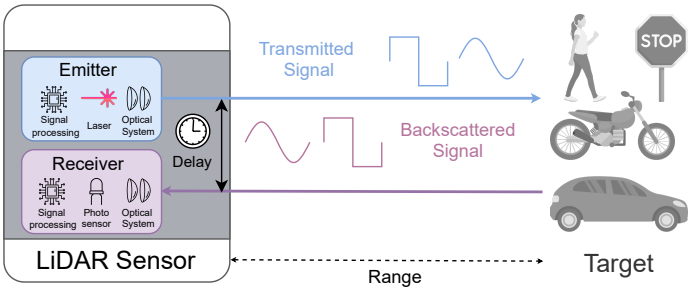


Figure 2. LiDAR Working Principle.

In a high-level overview, a LiDAR system is composed of two main components, a light Emitter (laser) and a Receiver (light detector), as depicted in Figure 2. The laser emits short light pulses with a well-defined time interval (few to several hundred nanoseconds), and with specific spectral properties into the optical steering system. By regulating mirror's angles, the system controls the direction of the light vertically and horizontally, providing multiple angle detection with just a single beam. Additionally, the optical properties of the beam can be changed by the lens system in order to achieve better performance ratios, e.g., with signal modulation schemes [22,30]. After reflecting into an object, the signal is reflected back to the sensor and the receiver collects the photons and it is followed by a system that, depending on the application, filters and selects specific wavelengths or polarization. The receiver system is also responsible to convert the optical signal into an electrical one and its intensity stored in a computing unit. The collected values are related to the photons time of travel and, consequently, the distance to the obstacle can be calculated. Within an automotive application, the main characteristics of a LiDAR sensor to be considered to include in the LiDAR testing and evaluation platform are: (1) horizontal and vertical field of view (FoV); (2) horizontal and vertical angular resolution (AR); (3) the influence of external illumination; (4) power consumption; and (5) sensor's minimum and maximum ranges. Such parameters are described as follows:

- The **Field of View** is one of the metrics that particularly defines the maximum angle a LiDAR sensor is able to detect objects, as shown in Figure 3. When two scanning angles are available, the sensor can scan over a 3D area defined by the Vertical FoV (VFoV) and the Horizontal FoV (HFoV). This test is designed to identify the maximum detection angles of the sensor to validate its defined values.

- The **Angular Resolution** defines the sensor's ability to better scan and detect objects within the FoV, as depicted in Figure 4. Higher resolutions allow for smaller blind spots between laser firings, enabling the detection of small objects and greater detail of the environment, particularly at higher detection ranges. Thus, this test is designed to identify the sensors angular resolution, both vertically and horizontally, in different areas of the FoV, verifying if the collected values match the requirements or sensor's characteristics defined by the manufacturer.

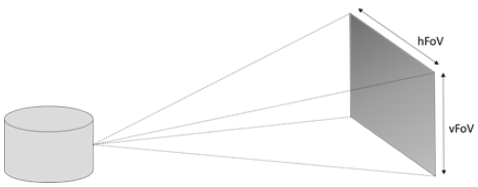


Figure 3. LiDAR horizontal and vertical FoV.

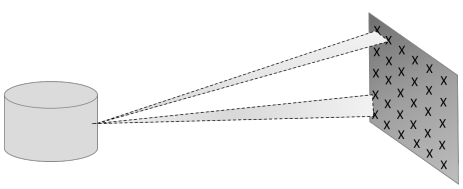


Figure 4. LiDAR horizontal and vertical AR.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2021 doi:10.20944/preprints202111.0562.v1

- **Background light & Sunlight** can have a severe impact on the sensor's behaviour. In real-world environments, LiDAR sensors can substantially decrease their performance when exposed to external light interference such as the sunlight backscattering in targets with high reflectivity characteristics. Removing such light noise can be particularly challenging due to solar radiation being a powerful light source present in a wide range of wavelengths [31]. Therefore, it is important to evaluate the sensor's output when exposed to background light in a controlled environment.

- The **Power Consumption** test aims at monitoring and analyzing the power consumption of the device under test (DUT) in different operation modes, configured parameters, and environment/target conditions.

- The **Range** can be defined as the minimum and the maximum distances in which the sensor successfully detects an object. While detecting the minimum range can be quite simple, finding the maximum range is not straightforward. This is dependent on the reflectivity of the target, which is considered detected when it appears in at least 90% (Detection Probability) of the measured frames in the point cloud. With a target reflectivity higher than 40-50%, detecting the maximum range in a straight line inside our testing laboratory (max. range of 100 meters) would be impossible for high-range sensors. However, by using the relationship between the returning signal strength from a specific target with a known reflectivity and the distance to the target, the maximum range for higher reflective targets can be deduced from the measurements performed with lower reflectivity ones. This method is based on the signal power arriving at the LiDAR detector as defined by the Equation 1, where A is a constant, R_{lab} is the target's reflectivity, and r_{lab}^2 is the target's distance.

$$P_{sig} = \frac{AR_{lab}}{r_{lab}^2} \tag{1}$$

  If the required minimum level for the returning signal remains the same regardless the target's reflectivity, the maximum distance (for any reflectivity value) can be calculated with the Equation 2, where R_{sim} is the target reflectivity to be simulated, and r_{sim} is the corresponding target distance calculated for the new reflectivity level. To reduce errors in the estimations, several measurements for the maximum range must be done, e.g., targets with reflectivities of 10%, 20%, and 40%.

$$r_{sim} = \sqrt{\frac{R_{sim} r_{lab}^2}{R_{lab}}} \tag{2}$$

3. LiDAR Evaluation and Testing

The Evaluation and Testing Platform for Automotive LiDAR Sensors aims at designing and developing a test-bench for LiDAR sensors (commercially available and Bosch prototypes currently under development) that is able to characterize and test the main parameters previously described in Section 2. Such tests are being performed inside two Bosch locations: (1) the Optical Lab (range up to 23 meters) and (2) the Long Range Measurements lab (range up to 100 meters).

3.1. System Architecture

The Optical Lab is composed of a set of equipment used to perform the desired tests. For the FoV, AR, and short-range measurements, we use a customized rail system and goniometric rotation system (RotGon) composed of a URS150BPP Rotation Stage and an M-BGM200BPP Goniometer from Newport. Regarding the power consumption, we use a direct current (DC) power analyzer (the N6705C 4 channel station), while for the external illumination influence we use an independent setup, which is further explained. Except for the backlight interference test, the testing and evaluation platform,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2021 doi:10.20944/preprints202111.0562.v1

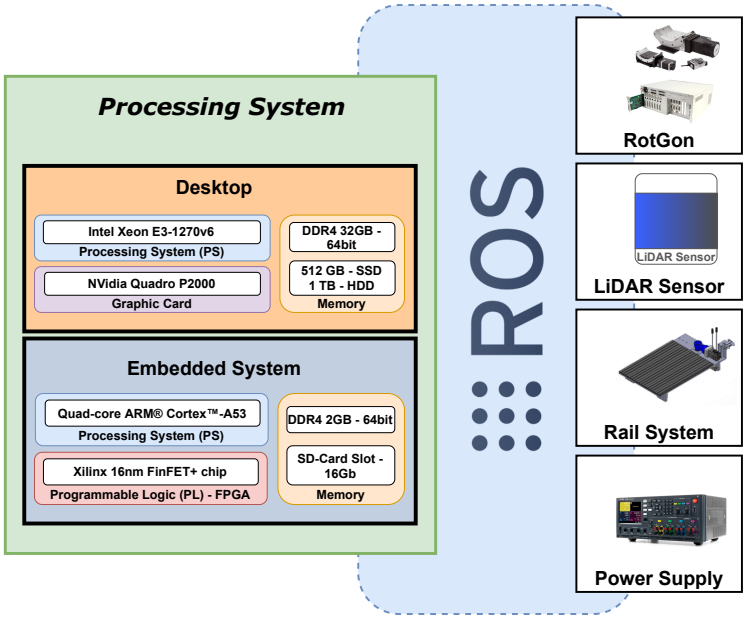


Figure 5. System Architecture.

whose architecture is depicted by Figure 5, connects all the equipment within a Robot Operating System (ROS) environment. All the processing tasks are distributed between a workstation and an embedded platform with acceleration capabilities through available field-programmable gate array (FPGA) technology. They both assure the complete system's functionality, allowing the laboratory to perform tests with either one of the systems alone or by both at the same time. The latter approach would also enable redundancy capabilities into the testing system.

The workstation is a great solution for developing the testing algorithms and other compute-intensive software tasks without concerning about the hardware resources. It is composed of a powerful desktop processor, a high-performance graphics card, and 32 gigabytes of ramdom-access memory (RAM). Within the workstation, some tasks, due to their heavy processing requirements, can either be performed by the available processing units or even by the combination of processors and the graphics card. Despite this solution, and having in mind the minimal setup and hardware resources, the testing sequences and algorithms are also supported by an embedded system built upon the Zynq UltraScale+ XCZU7EV-2FFVC1156 MPSoC (available in the ZCU104 Evaluation Kit). This MPSoC features a processing system (PS) that includes a quad-core Arm Cortex-A53 application processor, a dual-core Cortex-R5 real-time processor, a Mali-400 MP2 graphics processing unit, a 4KP60 capable H.264/H.265 video codec, programmable logic (PL) with FPGA technology, and 2GB of DDR4 memory. The embedded system allows exploring the available FPGA for accelerating heavy processing tasks, which can help in mitigating the overall processor's workload and avoid the utilization of the workstation. This can be useful in tests that require moving equipment. For the purpose of this article, all implementations were performed only on the workstation.

3.2. Lab Equipment

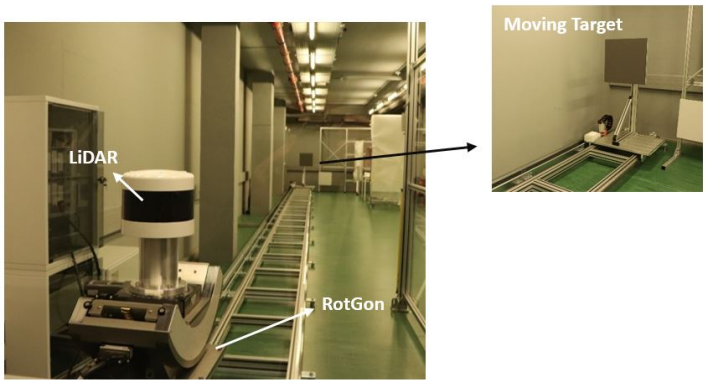


Figure 6. Evaluation and testing platform with the goniometric rotation system, a LiDAR sensor, and the rail system.

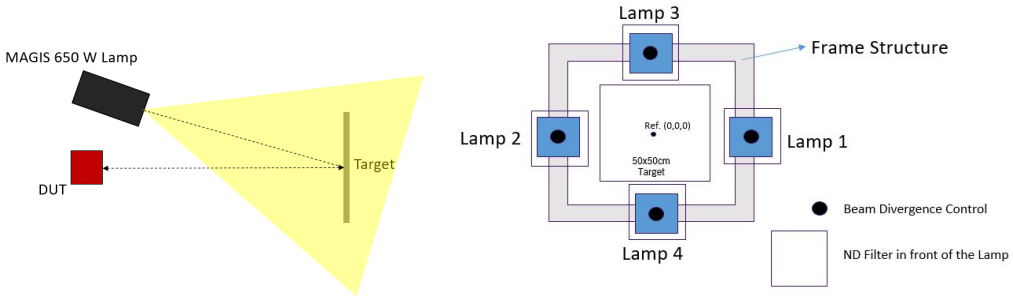
RotGon: The RotGon enables tilting/rotating the LiDAR sensors in three distinct angles. The rotation stage allows a continuous motion of 360º with a maximum speed of 40º/s and a resolution of 0.2 mdeg. The goniometer allows an angular range between -45º and 45º and features a worm mounted rotary encoder for improved accuracy and repeatability. Having a high precision, the RotGon is highly important for the measurement of the AR and the FoV.

Rail System: The rail system was designed to enable movements of a base that can handle weights up to 30 kg and that can be programmed by external communication. On top of the base, there will be targets of different reflectivity installed. The rail system's structure has a length of 25 meters and it is installed inside the laboratory. Figure 6 depicts the rail system with the LiDAR sensor installed on top of the RotGon (left side), and the moving platform with a mounted target at the end of the rail structure (right side). The rail system allows to control the velocity and the acceleration/deceleration of the moving target with given values in mm/s for velocity and $\pm\, mm/s^2$ for acceleration/deceleration. Prior to its utilization, the rail system was calibrated with a rangefinder equipment that was used for measuring several distances to a target with 95% reflectivity mounted on the rail system. The measurements were used as reference values for the internal position detector sensor.

Power Supply: This equipment is used to power and monitor the power consumption of the LiDAR sensor under test. The power supply used in the evaluation and testing platform is the N6705C DC Power Analyzer, which includes 4 independent channels that can be used to power and monitor 4 different connected modules. The voltage and current levels for each channel can be changed in real-time, allowing to further test the sensor's behaviour under different power source conditions.

External illumination influence (background & sunlight): This setup enables to test the influence of the external illumination by artificially changing the target's background light conditions. The main concept for this test is illustrated in Figure 7a. Its implementation, as depicted in Figure 7b, consists of four lamps attached to a metal frame (with a dimension of 50x50cm), which illuminate the target from four different positions to achieve a non-homogeneity of less than 10% in the center of the target. Considering that the frame is a two-dimensional plane, the lamps are placed on the center of each edge of the frame. The lamps used are the MAGIS 650 W Quartz-Halogen Fresnel Spotlight, based on a tungsten-quartz-halogen filament that can deliver 650W of power and allow the change of light divergence. Finally, to regulate the power applied to the target, a combination of different neutral density (ND) filters can be used in front of each lamp.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2021 doi:10.20944/preprints202111.0562.v1



(a) Main concept. **(b)** Frame with 4 MAGIS 650W spotlights.

Figure 7. Setup for the background light influence test.

3.3. ROS Software Architecture

The system's software stack is based on a ROS environment on top of a Linux operating system (OS), both supported by the embedded system and the workstation. Despite each distribution being different for each platform (due to the hardware resources asymmetry, processor architectures, etc.), the combination of Linux and ROS creates the required abstraction layer to develop software packages regardless their target platform. Alongside the required ROS core components, our software architecture is composed of eight software packages, as depicted by Figure 8.

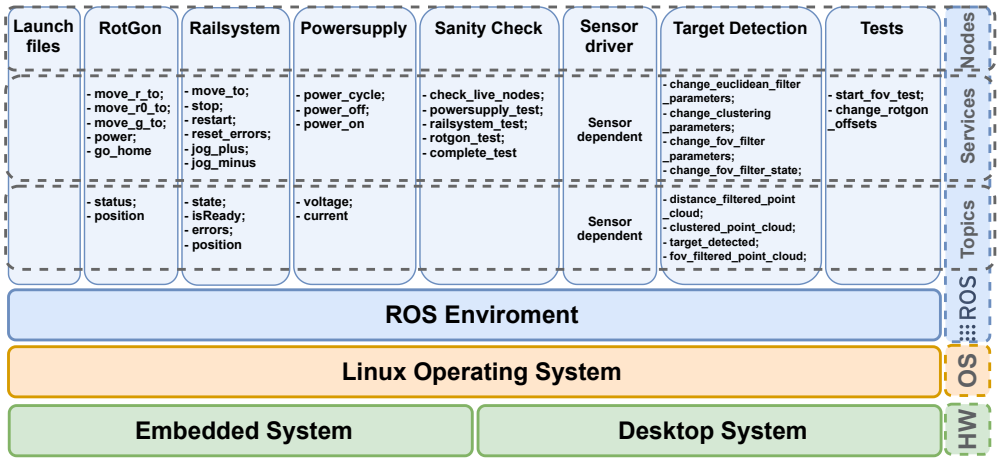


Figure 8. Software stack overview.

Launch files package: This package was developed to ease the system's launch with the correct testing setup. It allows flexible debug sessions with different LiDAR sensors, different sensor configurations, and several system setups where one or more components, e.g., RotGon, railsystem, may not be used. This package only presents launch files without services or topics available.

RotGon package: The RotGon component enables tilting/rotating the LiDAR sensors in three distinct angles. Therefore, this package provides three services (one for each axis), to move the sensor to the desired position/angle: ***move_r_to***, ***move_r0_to***, and ***move_g_to***. Additionally, it provides a self-reset service, ***go_home***, that moves all axis to the 0° degrees position, and a service, ***power***, to turn the power on and off. This package publishes information into two topics: one to display the current RotGon status regarding errors, ***status***, and another to output the current angle position in real-time, ***position***.

Railsystem package: This package is responsible for moving the target across the sensor's FoV. Therefore, it provides two services: one to send the target to a desired position, ***move_to***, and another to move the target at a constant speed and acceleration (***jog_plus***

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2021 doi:10.20944/preprints202111.0562.v1

for moving the target away from the sensor, and ***jog_minus*** for moving the target towards the sensor) until the service ***stop*** is called. Two more services are available to control the system regarding errors, ***reset_errors***, and communication issues, ***restart***. Like RotGon, this package has two topics, ***state*** and ***position***, to publish information about the testing equipment status and current position.

Powersupply package: The Powersupply package is responsible for controlling the sensor's power source. It provides three services to individually control each channel: one for turning on the power source, ***power_on***, one for turning off the power source, ***power_off***, and another one for resetting the power supply, ***power_cycle***. It can also set different voltage and current values, providing their real-time measurements present in the channel that is powering the sensor.

Sanity Check package: This package consists of a set of tools used to verify the full operation of the main system that are going to be used for testing a sensor, i.e., the rail system, the RotGon, and the power supply. It provides one service to individually test each core component, ***<equipment>_test***, one that tests the connectivity with the nodes, ***check_live_nodes***, and another that sequentially tests all the setup.

Sensor driver package: The sensor driver package is dependent on the sensor that is currently under test. Since most manufacturers provide a ROS-based driver with standard point cloud topics and services/launch files to interface their LiDAR sensors, the evaluation and testing platform can support a broad number of sensor drivers. Nonetheless, this package has to be manually installed and configured before changing the test configuration and the sensor.

Target detection package: This package is required for tests that depend on the target's visibility inside the sensor's FoV and consequently visible in the point cloud. It supports a set of services that are used to enable and configure several filters applied to the point cloud, such as target's distance, software-based FoV, etc. Such filters are further explained in the next Section. This service can output several topics with the filtered point clouds (one per filter), and one topic that continuously informs if the target is inside the sensor's FoV (***target_detected***).

Tests package: The Tests package contains the supported tests for the evaluation and testing platform that require the utilization of at least one of the equipment mentioned above. For each test, e.g., FoV and AR, a service is used to trigger the automated execution of the whole procedure. During the test, all the test outputs are saved in a ROS log file.

4. System Implementation

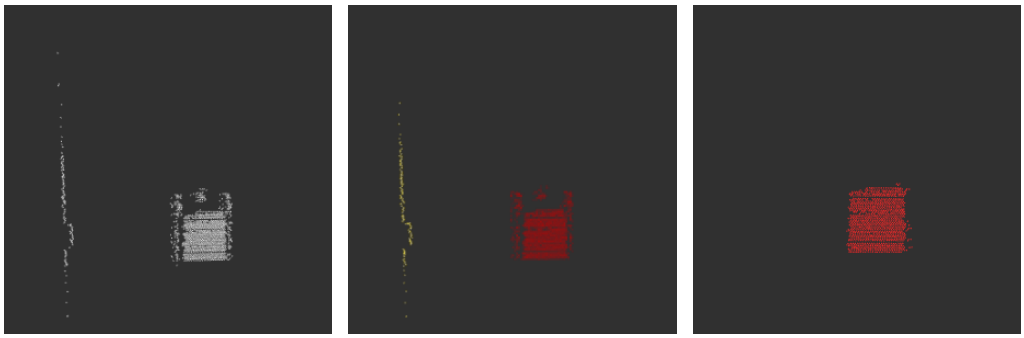
For the sake of simplicity, this Section only describes the software-based filters that can be applied to a point cloud, and the approaches used to calculate the FoV and the AR. The remaining tests, e.g., sensor's range, point cloud acquisition (with and without background illumination), are left aside of this article.

4.1. Point Cloud filtering for target detection

The evaluation and testing platform aims at supporting any COTS LiDAR sensor and Bosch prototypes under development. Regarding the supported tests, e.g., FoV, detecting targets can be challenging since not all sensors provide the point's intensity values along with the points coordinates information. Therefore, and in order to support all sensors' outputs, we have created a set of filters that can be used to detect targets without relying on the point's intensity values, such as a distance and a clustering filter.

Distance filter (DF): Since the target is placed at a well-known distance from the sensor, the output of this filter is a new point cloud (published to the ***filtered_point_cloud*** topic) containing points that are at this distance ± a threshold value (used to avoid removing

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2021 doi:10.20944/preprints202111.0562.v1



(a) Distance filter applied.

(b) Distance filter and euclidean clustering applied.

(c) Distance filter and euclidean clustering applied and tuned.

Figure 9. Target detection steps.

points that actually belong to the target). The result from applying the distance filter is shown in Figure 9a. This procedure not only removes undesired points, but also helps in reducing the computational costs of the subsequent tasks.

Cluster filter (CF): The cluster filter algorithm groups the points present in the point cloud and evaluates if the target is within the clusters created. Since the target's distance, size, and sensor's resolution can have an effect on the clustering results, this algorithm must be tuned afterwards. Figure 9b depicts the output of the CF without tuning its parameters, where two clusters were identified, as represented by the yellow and the red points. The points present inside the yellow cluster result from points that are at the same distance as the target, which must be removed on the next step. To detect if the resulting clusters represent the target, an euclidean clustering filter is applied. Since the point density within the target's cluster is higher than in other objects at the same distance, this filter analyzes the neighbour points of each point within a defined search radius $R1$. If a neighbor point is inside this search radius $R1$, it belongs to the same cluster and it is kept on the point cloud. Otherwise, it is removed. This task is performed by resorting to the method *EuclideanClusterExtraction.extract* present in the point cloud library (PCL) [32]. The parameters used to configure this method are:

- ***Cluster Tolerance:*** Defines the search radius $R1$. If the chosen value is to small, the same target can be divided into multiple clusters. On the other hand, if this value is to high, multiple objects can be set as just one cluster. This parameter allows an interval value between 0.01 and 1 meter.
- ***Minimum Cluster Size:*** This parameter is used to define the minimum number of points required to form a cluster. It allows values between 1 and 10 000 points.
- ***Maximum Cluster Size:*** This parameter defines the maximum number of points used to form a cluster. It supports a minimum of 2 and a maximum of 50 000 points.

Figure 9c depicts the point cloud output after applying the tuned euclidean clustering filter. When the target's cluster is found, this filter also publishes a message to the ***target_detected*** topic using the ***TargetInfo*** message type, which basically contains a boolean variable (True if the target is being detected and False otherwise), and the number of points inside the cluster. The new point cloud that only contains the target's cluster is published in the *clustered_point_cloud* topic, which can finally be used in testing the sensor's parameters, e.g., the FoV and the AR.

FoV software filter (FoVSF): The purpose of this filter is to enable support for any LiDAR sensor on the market, including rotating-based COTS LiDAR sensors widely used in automotive applications, which usually provide a 360º horizontal FoV. Notwithstanding, for the purpose of testing and validating the platform, which must also support LiDAR sensors with limited FoV, this software filter allows the cropping of the point cloud to a desired horizontal and vertical FoV. The filter runs in two steps. First, it converts the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2021 doi:10.20944/preprints202111.0562.v1

points in the point cloud from the cartesian coordinate system to the spherical one by using Equation 3 for calculating the azimuth, and Equation 4 for the elevation angle. In the second step, the algorithm discards the points from the point cloud that are not within the desired thresholds. The output of this filter is a ROS topic with a new point cloud containing the points that are within the configured FoV. Later in Section 5, it is possible to see an application of this filter.

$$\theta = \begin{cases} \frac{\arctan\frac{y}{x}\times 180}{\pi}, & \text{if } x \geq 0 \text{ and } y \geq 0 \\ \frac{\arctan\frac{y}{x}\times 180}{\pi} + 180, & \text{if } x < 0 \text{ and } y \geq 0 \\ 270 - \frac{\arctan\frac{y}{x}\times 180}{\pi}, & \text{if } x < 0 \text{ and } y < 0 \\ \frac{\arctan\frac{y}{x}\times 180}{\pi} + 360, & \text{otherwise} \end{cases} \tag{3}$$

$$\varphi = \begin{cases} 90 - \frac{\arctan\frac{\sqrt{x^2+y^2}}{z}\times 180}{\pi}, & \text{if } z > 0 \\ -\left(\frac{\arctan\frac{\sqrt{x^2+y^2}}{z}\times 180}{\pi} + 90\right), & \text{if } z < 0 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

4.2. Implementation of the FoV test

The test to determine the sensor's FoV consists of using a target with a well-known size and reflectivity, placed at a know distance on top of the rail system target's holder. Since the rail system can only provide variable ranges, we can take advantage of the RotGon to move the sensor both in the horizontal and vertical directions and check when the target moves outside the sensor's FoV. By using the position data from the RotGon, it is possible to determinate the sensor's FoV. This procedure is illustrated in Figure 10.

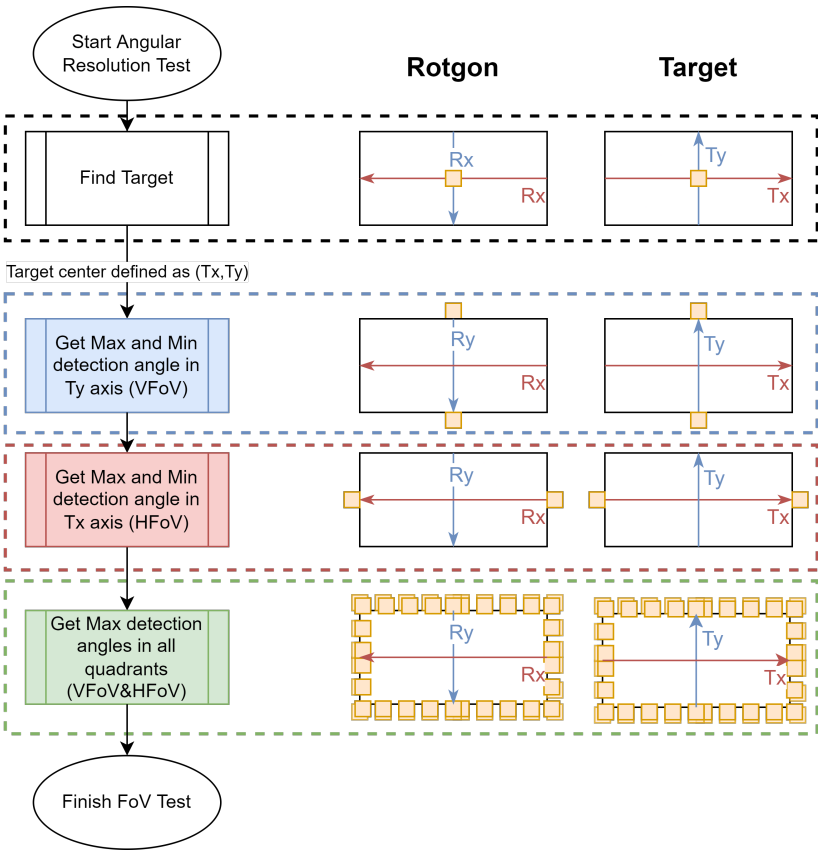


Figure 10. Field of View flowchart Overview.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2021 doi:10.20944/preprints202111.0562.v1

The test starts with a routine that uses the services provided by the ***Target Detection*** ROS package (previously explained) to find a target inside the sensor's point cloud data. If the target is detected, the algorithm starts measuring the FoV. Firstly, it starts by finding the maximum and minimum angles in the vertical axis for achieving the vertical FoV (blue color). Next, the same concept is applied to the horizontal axis for finding the horizontal FoV (red color). Finally, the values retrieved in the previous tasks are used as the starting conditions to test the consistency of both horizontal and vertical FoV in the limits of all quadrants (green color).

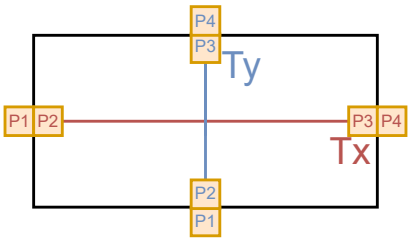


Figure 11. Target positions for the FoV measurement.

$$
\begin{aligned}
FoV_1 &= P1 - P3 \\
FoV_2 &= P2 - P4 \\
FoV &= \frac{FoV_1 + FoV_2}{2} \\
FoV_{min} &= \frac{P1 + P2}{2} \\
FoV_{max} &= \frac{P3 + P4}{2}
\end{aligned}
\tag{5}
$$

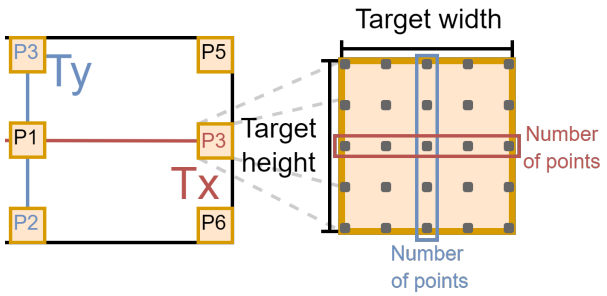
In all procedures, to get the maximum and minimum detection angles, the system increments/decrements the RotGon angles until the target reaches the four different positions illustrated in Figure 11:

- *P1* - Last position where the target is completely outside of the FoV;
- *P2* - First position where the target is completely inside of the FoV;
- *P3* - Last position where the target is completely inside of the FoV;
- *P4* - First position where the target is completely outside of the FoV after *P3*;

Then, based on Equations 5, for each axis it is calculated the minimum detection angle (FoV_{min}), maximum detection angle (FoV_{max}) and the FoV. Since the method uses the mean values of two known positions to achieve the minimum and maximum values, the target size is automatically removed from the calculations. Moreover, and in order to calculate the FoV as close as possible to its real value, in the limits of the FoV (where the target starts to disappear) we use the lowest angular step provided by the RotGon, which is 0.01°.

4.3. Implementation of the AR test

The sensor's AR defines the distance (in degrees) between two consecutive measured points. A smaller distance represents a better sensor's AR, as well as a higher number of collected points per frame. With this concept in mind, the most straightforward way to calculate the angular resolution of a LiDAR output is by firstly counting the number of points present in the point cloud, obtained from a high-reflectivity target with a known size (T_{width} x T_{height}) (Figure 12), placed at a known distance (T_{dist}), and later, converting the value to angular resolution using Equations 6.



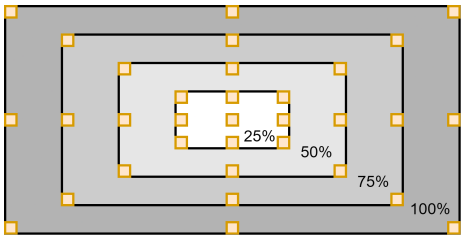
$$
\begin{aligned}
AR_h &= \frac{2\arcsin\left(\frac{T_{width}}{2T_{dist}}\right)}{H_{number\_points}} \\
AR_v &= \frac{2\arcsin\left(\frac{T_{height}}{2T_{dist}}\right)}{V_{number\_points}}
\end{aligned}
\tag{6}
$$

Figure 12. Points reflected by a known target.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2021 doi:10.20944/preprints202111.0562.v1

Based on those trigonometric functions which relate the right-angled triangle created by half the size of the target ($\frac{T_{width}}{2}$ or $\frac{T_{height}}{2}$), and the distance between the sensor and the target (T_{dist}), it is possible to measure the angle needed to perceive half of the target. Then, the angular resolution can be achieved by dividing this angle by half the number of points that represent the target. Like in the FoV, the AR test also re-measures the vertical and horizontal angular resolutions at different target positions. However, such positions are not the same as the FoV. While for the FoV, the goal was to get the RotGon angles where the target is entirely inside or outside the point cloud; for the AR test, the goal is to get multiple positions, as illustrated by Figure 13, where the angular resolution can be different.

Moreover, and since a sensor usually presents more point density in the center of the point cloud, the AR can vary several tenths of degrees depending on the target position. Hence, this measurement is performed within different regions inside the point cloud defined by a software FoV filter that reduces the FoV by 25% into smaller FoV areas, as depicted by Figure 13. After computing both vertical and horizontal AR for each target position, the test summarizes the information by calculating the arithmetic mean of each virtual FoV based on Equations 7. For each percentage of FoV, nine positions are considered: the central position that is common for all virtual FoV and all the eight border positions.



$$AR_{h_{a\%}} = \frac{1}{9}\left(\sum_{i=1}^{9} AR_{h_{Pia\%}}\right)$$
$$AR_{v_{a\%}} = \frac{1}{9}\left(\sum_{i=1}^{9} AR_{v_{Pia\%}}\right) \quad (7)$$

Figure 13. Target positions for a full AR evaluation.

5. Results

To validate the evaluation and testing platform and the algorithms developed for testing LiDAR sensors, and for the sake of simplicity, this Section only shows the setup that we have created for testing the FoV of a COTS LiDAR sensor. For this purpose we have selected the Velodyne VLS-128, which is one of the highest resolution sensor available in the market specially designed for autonomous vehicles, with the setup previously depicted in Figure 6. This way, this simple test can show the full functionalities of the system since it uses most of the equipment available inside the laboratory (the RotGon, the rail system, and the power supply), and the software point cloud filtering modules previously discussed (DF, CF, and FoVSF). To provide reliable, precise and accurate measurements, all equipment was previously calibrated, and to ensure the proper operation of the evaluation and testing platform, we run a sanity check sequence before testing the sensor.

5.1. Sanity check

The sanity check sequence tests independently the power supply, the rail system, and the RotGon. This is provided by the ***Sanity Check*** package, which provides four main services: ***powersupply_test***, ***railsystem_test***, ***rotgon_test*** and ***complete_test***. Before running the sanity check procedure each service uses the ***check_live_nodes*** service that checks if the ROS node corresponding to the equipment being tested is turned on and visible within the ROS network. Regarding the outcome of the sanity checks, the test either results in **success**, meaning that the system is ready to test the LiDAR sensor, or in **failure**, reporting which component resulted on an error. Those errors reported by each equipment's node are summarized in Table 1.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2021 doi:10.20944/preprints202111.0562.v1

Table 1: Sanity check error list.

Equipment	Result	Description
Powersupply	1	No problems detected
	2	No node detected in the ROS Environment
	3	Values detected not matching the expected values
Railsystem	1	No problems detected
	2	No node detected in the ROS Environment
	3	Component not ready
	4	Component has internal errors
	5	Component not moving after a moving command
	6	Component not stopping after a stop command
	7	Component not in the correct position
Rotgon	1	No problems detected
	2	No node detected in the ROS Environment
	3	Component not in the correct position

Power supply sanity check: After checking if the *powersupply_node* is alive, this test verifies if any sensor is connected to the system by getting the list of connected sensors. Then, it evaluates if each connected sensor's parameters match the values reported by the power supply. There are three possible outcomes: (1) the node is unresponsive; (2) the connected sensor matches the configured parameters; or (3) the power supply readings do not match the expected values.

Rail system sanity check: Similarly to the power supply, the rail system routine begins by testing if its corresponding ROS node is alive. Next, the rail system is validated by sending the platform that holds the target into different positions while checking the system's response. Therefore, two dedicated services are defined: ***move_sequence*** and ***is_moving***. While the first is responsible for calling the ***move_to*** services, the latter checks if the target is, in fact, moving or at the desired position. The rail system sanity check has six possible results: (1) no problems were detected; (2) the node is unresponsive; (3) the node's internal flags indicate a busy state, i.e., the rail system is not ready to receive commands; (4) the internal flags indicate internal error status; (5) the target did not move after a ***move_to*** command; (6) the target could not stop after a stop command; and (7) the target is not at the expected position.

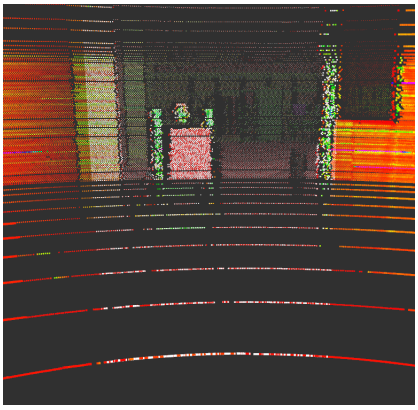
RotGon sanity check: This routine verifies four moving commands: ***go_home***, ***move_r_to***, ***move_r0_to*** and ***move_g_to***. Next it tests if the moving parts (one for each axis) is at the desired angle. This test can report three possible situations: (1) the RotGon is alive and running; (2) The RotGon node is unresponsive; and (3) the RotGon positions are different from the expected.

5.2. FoV Test

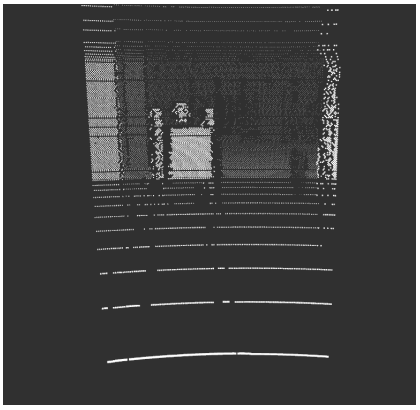
To validate the FoV testing algorithm, we have used different FoV values within the range of the Velodyne VLS-128: horizontal FoV of 360º, and vertical FoV of 40º. This can be adjusted by using FoVSF provided by the ***Target Detection*** package. It is important to mention that we have forced this step to prove the functionality of the FoVSF (mostly for the horizontal plane since the VLS-128 provides a 360º horizontal FoV), which may not be required when testing sensors with limited FoV values and it is required to validate the parameters provided and set by the manufacturer. After placing the target at a known distance and within the visibility of the configured FoV, the DF is applied to remove the points in the point cloud that are outside the desired range, resulting in a cleaner point cloud that will help in reducing the computational requirements of the subsequent tasks. Lastly, the CF step is applied. From here, the system has successfully locked the target and is finally able to evaluate the FoV value that is known and was previously set.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 30 November 2021 doi:10.20944/preprints202111.0562.v1

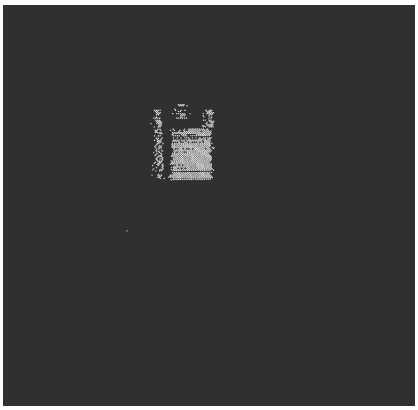
The results are published in real-time to the ***target_detected*** topic that is being subscribed by the running test script, which in this case corresponds to the FoV test. Figure 14 depicts all the performed steps in order to detect and lock the target in the point cloud: (1) Figure 14a shows the raw data sent by the VLS-128; (2) Figure 14b depicts the FoVSF being applied; (3) Figure 14c illustrates the DF output; and (4) Figure 14d shows only the point cluster that corresponds to the target visible and locked in the point cloud.



(a) VLS-128 point cloud (raw data)



(b) FoVSF applied



(c) DF applied

(d) DF and CF applied

Figure 14. Target detection steps.

Hereafter, we run the ***Tests*** package, which is responsible to perform the algorithms previously described in Figures 10 and 11. The gathered results are summarized in Table 3, which were obtained using the parameters described in Table 2.

Table 2: Filter parameters.

Cluster filter			Distance filter		Target
Cluster Tolerance	Cluster Size (min)	Cluster Size (max)	Threshold (min)	Threshold (max)	Distance
0.03 m	100 pt	2000 pt	5.5 m	5.6 m	5.5 m

We have performed three distinct tests, Test 1, Test 2, and Test 3, which consisted in changing the sensor's FoV and validating the configured values. For the vertical FoV, we could measure different regions inside the original sensor's values (40º) since this area is in the range of the RotGon's rotation angles. For the horizontal FoV, we have used three different values: (1) 60º; (2) 55º; and (3) 135º.

Table 3: Test results.

	FoV filter				Measured FoV			
	Horizontal FoV (min)	Horizontal FoV (max)	Vertical FoV(min)	Vertical FoV (max)	Horizontal FoV (min)	Horizontal FoV (max)	Vertical FoV (min)	Vertical FoV (max)
Test 1	0º	60º	-10º	15º	0.03º	59.97º	-9.95º	14.89º
Test 2	20º	75º	-25º	0º	20.02º	74.89º	-24.69º	-0.13º
Test 3	0º	135º	0º	17º	-0.08º	134.7º	0.04º	16.92º

In all results we could see the proper operation of the evaluation and testing platform, where some calculated angles have slight deviations from the desired values. It is important to mention that our measurements are being performed from a sensor's receiver perspective, which are only based on the point cloud data provided by the sensor and which are going to be used by other (high-level) applications within the perception system of the car. Therefore, we consider that at this order of magnitude, these deviations are not critical and can still validate the sensors parameters being tested. When a more accurate analysis is required, we can also submit the sensor to an end-of-line testing scenario that is part of the laboratory responsible to test sensors under development within other Bosch projects. However, end-of-line testing for the laser transmitter are out of the scope of this article.

6. Conclusions

This article presents an evaluation and testing platform that is able to test and validate different parameters of LiDAR sensors designed for automotive applications. The platform was built upon a set of equipment supported by a ROS software environment. Since the purpose of this platform is to evaluate any LiDAR sensor available in the market, we have created several ROS packages to control and automate the tests, and a set of software-based filters for being able to support any sensor's output based only on point cloud data information. Despite all tests being performed from the sensor's receiver perspective, the results are quite promising. We could validated the output of a Velodyne VLS-128 sensor, as well as the concept of our point cloud filtering approaches such as the FoV, distance, and point clustering. Hereafter, future developments will provide the support for tests under real-life environments, such as adverse weather.

Acknowledgments: This work is supported by: European Structural and Investment Funds in the FEDER component, through the Operational Competitiveness and Internationalization Programme (COMPETE 2020) [Project nº 037902; Funding Reference: POCI-01-0247-FEDER-037902].

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Daily, M.; Medasani, S.; Behringer, R.; Trivedi, M. Self-Driving Cars. *Computer* **2017**, *50*, 18–23.
2. Badue, C.; Guidolini, R.; Carneiro, R.V.; Azevedo, P.; Cardoso, V.B.; Forechi, A.; Jesus, L.; Berriel, R.; Paixão, T.M.; Mutz, F.; de Paula Veronese, L.; Oliveira-Santos, T.; De Souza, A.F. Self-driving cars: A survey. *Expert Systems with Applications* **2021**, *165*, 113816.
3. Gao, C.; Wang, G.; Shi, W.; Wang, Z.; Chen, Y. Autonomous Driving Security: State of the Art and Challenges. *IEEE Internet of Things Journal* **2021**, pp. 1–1.
4. Yurtsever, E.; Lambert, J.; Carballo, A.; Takeda, K. A Survey of Autonomous Driving: Common Practices and Emerging Technologies. *IEEE Access* **2020**, *8*, 58443–58469.
5. Litman, T. *Autonomous vehicle implementation predictions*; Victoria Transport Policy Institute Victoria, Canada, 2021.
6. Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles. Sae recommended practice, SAE International, Geneva, CH, 2021.
7. Guerrero-Ibáñez, J.; Zeadally, S.; Contreras-Castillo, J. Sensor Technologies for Intelligent Transportation Systems. *Sensors (Basel, Switzerland)* **2018**, *18*, 1212.
8. Marti, E.; de Miguel, M.A.; Garcia, F.; Perez, J. A Review of Sensor Technologies for Perception in Automated Driving. *IEEE Intelligent Transportation Systems Magazine* **2019**, *11*, 94–108.
9. Shahian Jahromi, B.; Tulabandhula, T.; Cetin, S. Real-Time Hybrid Multi-Sensor Fusion Framework for Perception in Autonomous Vehicles. *Sensors* **2019**, *19*.

10. Mohammed, A.S.; Amamou, A.; Ayevide, F.K.; Kelouwani, S.; Agbossou, K.; Zioui, N. The Perception System of Intelligent Ground Vehicles in All Weather Conditions: A Systematic Literature Review. *Sensors* **2020**, *20*.
11. Warren, M.E. Automotive LIDAR Technology. 2019 Symposium on VLSI Circuits, 2019, pp. C254–C255.
12. Li, Y.; Ibanez-Guzman, J. Lidar for Autonomous Driving: The Principles, Challenges, and Trends for Automotive Lidar and Perception Systems. *IEEE Signal Process. Mag.* **2020**, *37*, 50–61.
13. Roriz, R.; Cabral, J.; Gomes, T. Automotive LiDAR Technology: A Survey. *IEEE Transactions on Intelligent Transportation Systems* **2021**, pp. 1–16.
14. Arnold, E.; Al-Jarrah, O.Y.; Dianati, M.; Fallah, S.; Oxtoby, D.; Mouzakitis, A. A Survey on 3D Object Detection Methods for Autonomous Driving Applications. *IEEE Transactions on Intelligent Transportation Systems* **2019**, *20*, 3782–3795.
15. Shi, S.; Wang, X.; Li, H. PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. *2019 IEEE/CVF Conf. on Comput. Vision and Pattern Recognition (CVPR)* **2019**, pp. 770–779.
16. Wu, J.; Xu, H.; Tian, Y.; Pi, R.; Yue, R. Vehicle Detection under Adverse Weather from Roadside LiDAR Data. *Sensors* **2020**, *20*.
17. Wang, H.; Wang, B.; Liu, B.; Meng, X.; Yang, G. Pedestrian recognition and tracking using 3D LiDAR for autonomous vehicle. *Robotics and Autonomous Systems* **2017**, *88*, 71–78.
18. Peng, X.; Shan, J. Detection and Tracking of Pedestrians Using Doppler LiDAR. *Remote Sensing* **2021**, *13*.
19. Huang, W.; Liang, H.; Lin, L.; Wang, Z.; Wang, S.; Yu, B.; Niu, R. A Fast Point Cloud Ground Segmentation Approach Based on Coarse-To-Fine Markov Random Field. *IEEE Trans. on Intell. Transp. Syst.* **2021**, pp. 1–14.
20. Karlsson, R.; Wong, D.R.; Kawabata, K.; Thompson, S.; Sakai, N. Probabilistic Rainfall Estimation from Automotive Lidar, 2021, [arXiv:eess.SP/2104.11467].
21. Raj, T.; Hashim, F.; Huddin, B.; Ibrahim, M.; Hussain, A. A Survey on LiDAR Scanning Mechanisms. *Electronics* **2020**, *9*.
22. Behroozpour, B.; Sandborn, P.A.M.; Wu, M.C.; Boser, B.E. Lidar System Architectures and Circuits. *IEEE Communications Magazine* **2017**, *55*, 135–142.
23. Jiménez, J. Laser diode reliability: Crystal defects and degradation modes. *Comptes Rendus Physique* **2003**, *4*.
24. 2-D Optical-CDMA modulation in automotive time-of-flight LIDAR systems. 2020, Vol. 2020-July.
25. A CDMA Modulation Technique for Automotive Time-of-Flight LiDAR Systems. *IEEE Sensors Journal* **2017**, *17*, 3507–3516.
26. AVM / LiDAR sensor based lane marking detection method for automated driving on complex urban roads. 2017.
27. Jokela, M.; Kutila, M.; Pyykönen, P. Testing and Validation of Automotive Point-Cloud Sensors in Adverse Weather Conditions. *Applied Sciences* **2019**, *9*.
28. Vargas Rivero, J.R.; Gerbich, T.; Teiluf, V.; Buschardt, B.; Chen, J. Weather Classification Using an Automotive LIDAR Sensor Based on Detections on Asphalt and Atmosphere. *Sensors* **2020**, *20*.
29. Chan, P.H.; Dhadyalla, G.; Donzella, V. A Framework to Analyze Noise Factors of Automotive Perception Sensors. *IEEE Sensors Letters* **2020**, *4*, 1–4.
30. Suss, A.; Rochus, V.; Rosmeulen, M.; Rottenberg, X. Benchmarking time-of-flight based depth measurement techniques. Smart Photonic and Optoelectronic Integrated Circuits XVIII; He, S.; Lee, E.H.; Eldada, L.A., Eds. International Society for Optics and Photonics, SPIE, 2016, Vol. 9751, pp. 199 – 217.
31. Sun, W.; Hu, Y.; MacDonnell, D.G.; Weimer, C.; Baize, R.R. Technique to separate lidar signal and sunlight. *Opt. Express* **2016**, *24*, 12949–12954.
32. Rusu, R.B.; Cousins, S. 3D is here: Point Cloud Library (PCL). 2011 IEEE International Conference on Robotics and Automation (ICRA), 2011, pp. 1–4.