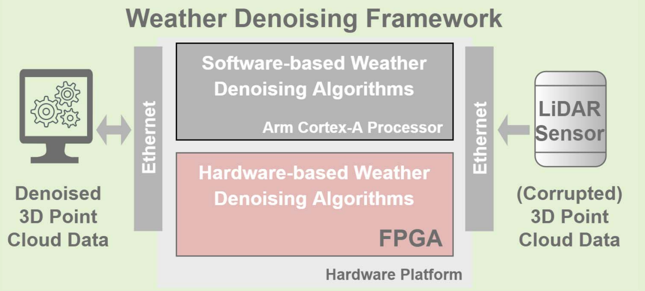
DIOR: A Hardware-Assisted Weather Denoising Solution for LiDAR Point Clouds

Ricardo Roriz, André Campos, Sandro Pinto, and Tiago Gomes

***Abstract*—The interest in developing and deploying fully autonomous vehicles on our public roads has come to a full swing. Driverless capabilities, widely spread in modern vehicles through advanced driver-assistance systems (ADAS), require highly reliable perception features to navigate the environment, being light detection and ranging (LiDAR) sensors a key instrument in detecting the distance and speed of nearby obstacles and in providing high-resolution 3D representations of the surroundings in real-time. However, and despite being assumed as a game-changer in the autonomous driving paradigm, LiDAR sensors can be very sensitive to adverse weather conditions, which can severely affect the vehicle's perception system behavior. Aiming at improving the LiDAR operation in challenging weather conditions, which contributes to achieving higher driving automation levels defined by the Society of Automotive Engineers (SAE), this article proposes a weather denoising method called Dynamic light-Intensity Outlier Removal (DIOR). DIOR combines two approaches of the state-of-the-art, the dynamic radius outlier removal (DROR) and the low-intensity outlier removal (LIOR) algorithms, supported by an embedded reconfigurable hardware platform. By resorting to field-programmable gate array (FPGA) technology, DIOR can outperform state-of-the-art outlier removal solutions, achieving better accuracy and performance while guaranteeing the real-time requirements.**


Weather Denoising Framework
Software-based Weather Denoising Algorithms
Arm Cortex-A Processor
Hardware-based Weather Denoising Algorithms
FPGA
Hardware Platform
Ethernet
Ethernet
LiDAR Sensor
Denoised 3D Point Cloud Data
(Corrupted) 3D Point Cloud Data

***Index Terms*—ADAS, autonomous vehicles, FPGA, LiDAR, point cloud filtering, weather denoising.**

I. Introduction

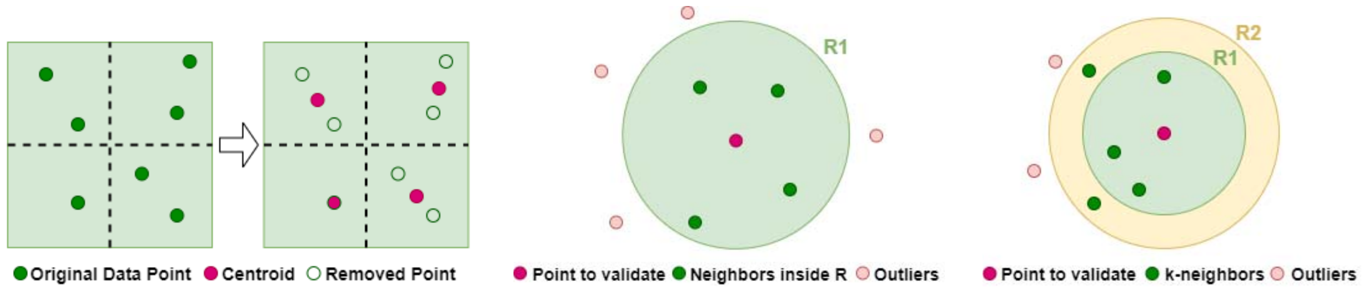
NOWADAYS, and more than a decade after the first self-driving car winning the DARPA Challenge, the interest in developing and deploying fully autonomous vehicles has come to a full swing. An autonomous vehicle requires reliable solutions to provide an accurate mapping of the surroundings, which is only possible with multi-sensor perception systems relying on a combination of Radar, Cameras, and light detection and ranging (LiDAR) sensors [1]–[4]. Working together, they provide the ability to detect the distance and speed of nearby obstacles as well as their aspect to safely navigate the environment, contributing to different Society of Automotive Engineers (SAE) Levels of driving automation. While levels 0, 1, and 2 require the driver to monitor the surroundings, with higher levels the automated system monitors the entire driving environment. The utilization of LiDAR sensors in the automotive sector is relatively new, but already assumed as the key technology towards full driverless vehicles, since they can provide high-resolution 3D representations of the surroundings in real-time [5]–[7]. A LiDAR sensor works by illuminating a target with an optical pulse and measuring the characteristics of the return signal, where the target's distance is obtained by calculating the round-trip delay of the reflected light. Despite simple, applying this principle is not straightforward since this technology is quite sensitive to several external disturbances.

The advances around LiDAR keep improving its measuring techniques and imaging architectures [6], [8]. Measuring techniques focus in obtaining the distance by calculating the time-of-flight (ToF) of the emitted light through pulsed or continuous wave approaches based on signal modulation to improve the immunity and increase the signal-to-noise ratio (SNR). Such techniques can be deployed on a variety of imaging architectures, which range from beam steering sensors that use a rotor-based mechanical part to scan the environment to solid-state sensors with less or no moving parts [6], [8]. LiDAR technology is steadily improving and being applied to a wide range of applications. Accurate and precise measurements of the surroundings through a 3D point cloud representation can assist the perception systems in several tasks [3], e.g., obstacles, objects, and vehicles detection [9]–[11]; pedestrians recognition and tracking [12]; ground segmentation for road filtering [13]; among others [14]. Nonetheless, the sparse 3D point cloud of a LiDAR sensor can

Manuscript received November 10, 2021; accepted December 6, 2021. Date of publication December 8, 2021; date of current version January 12, 2022. This work was supported by the Fundação para a Ciência e Tecnologia (FCT) within the Research and Development Units Project Scope under Grant UIDB/00319/2020. The associate editor coordinating the review of this article and approving it for publication was Prof. Piotr J. Samczynski. *(Corresponding author: Tiago Gomes.)*

The authors are with the Centro ALGORITMI, Universidade do Minho, Campus Azurém, 4800-058 Guimarães, Portugal (e-mail: mr.gomes@dei.uminho.pt).

Digital Object Identifier 10.1109/JSEN.2021.3133873


● Original Data Point ● Centroid ○ Removed Point
R1
● Point to validate ● Neighbors inside R ● Outliers
R2
R1
● Point to validate ● k-neighbors ● Outliers

(a) Voxel Grid. (b) ROR, DROR, and LIOR. (c) SOR.

Fig. 1. Weather denoising methods based on neighbor points relationships.

be subject to several noise sources, e.g., internal components, mutual interference, reflectivity issues, light, and adverse weather, which can corrupt the measurements and the output data.

Most of the current data processing algorithms usually assume ideal weather. However, it is known that adverse conditions can affect the normal operation of the vehicle's perception system. This problem has been thoroughly addressed in the literature [4], [15]–[17], where several studies have benchmarked the sensor's response under different and extreme weather conditions such as fog [18]–[22], rain [18], [19], [23]–[26], and snow [11], [27]. All have evaluated how adverse weather can affect the vehicle's behavior and proposed different mitigation approaches. Current state-of-the-art solutions include: (i) simulators to analyze the influence of adverse weather under different road conditions and scenarios [21], [23]–[26]; (ii) improved background filtering and object clustering methods to better process the roadside LiDAR data [11], [22]; (iii) learning approaches based on convolutional neural networks (CNNs) [19], [21]; and (iv) weather classification systems [17], [18], [22]. The latter use the information provided by the LiDAR to predict weather and adapt the sensor's operation based on atmosphere and asphalt changes, which contributes to a better understanding of the surroundings according to the actual environmental conditions. Despite the broad research that has been devoted to tackling problems caused by adverse weather, it is crucial to provide solutions that can overcome current limitations related to the performance, accuracy, and overall system complexity.

With this work, we enrich the state-of-the-art with: (i) a new weather denoising method called DIOR, which combines two approaches of state-of-the-art algorithms; (ii) a weather denoising framework for LiDAR point cloud data that supports state-of-the-art algorithms assisted by a reconfigurable hardware platform; and (iii) an extended benchmark and comparative review between DIOR and current approaches.

II. Point Cloud Weather Denoising

The LiDAR point cloud can be differently affected due to specific weather conditions. While the water droplets present in rain and fog scatter the emitted light (reducing the operating range and producing inaccurate measurements), the solid particles present in smoke and snow can originate ghost information in the point cloud. With the most important metrics for a LiDAR weather denoising system in mind, such as accuracy, performance, and algorithm's simplicity, state-of-the-art solutions include voxel grid (VG) filters [28] and algorithms based on outlier removal techniques, e.g., radius outlier removal (ROR) and dynamic radius outlier removal (DROR) [27], statistical outlier removal (SOR) and fast cluster statistical outlier removal (FCSOR) [29], and low-intensity outlier removal (LIOR) [30]. An outlier is an observation that is noticeably different from other (adjacent) observations in a dataset. Regarding the 3D point cloud data, which can hold millions of generated points per second, there are several points that do not share any relation or characteristics, e.g., distance or intensity, with their neighbors. Such outlier points mostly represent noise in the point cloud, which can compromise the normal operation of the sensor or affect high-level applications such as object detection and classification algorithms. Aside from outlier removal methods, learning-based denoising algorithms also started to emerge [19], [21]. However, they are considered complex since they require real-world datasets and powerful computational resources. Thus, they are out of the scope of this work.

***VG*-based methods**, (Fig. 1a), consist in defining 3D boxes (forming a voxel grid) in the 3D space of the point cloud. Then, for each voxel, the algorithm selects a point (usually the central point or the centroid of the box) to approximate the remaining points inside the voxel. Because of this feature, VGs can be used for point cloud denoising since a noise point often lacks in neighbors or does not share information with them, which will end in being removed from the point cloud. VG methods are fast and relatively simple to implement. However, because they cause the down-sampling of the information within the voxel, not only noise points will be removed from the point cloud but also points with useful information about the surroundings.

The ROR method, depicted in Fig. 1b, consists of a simple technique that computes the distance of each point to its neighbors (on a k-d tree data structure) within a fixed search radius ***R1***. When the number of neighbor points is below the defined threshold, the point is classified as an outlier. Despite being simple to implement, the ROR algorithm does not perform well for distant objects in the 3D LiDAR point cloud due to variation in the point distances, resulting in points wrongly classified as outliers. To address this limitation, DROR (Fig. 1b) rather than using static search, dynamically increases the searching radius ***R1*** as the distance to the measured points also increases [27]. DROR was evaluated and compared with other state-of-the-art algorithms, e.g., ROR and VG, available in the Point Cloud Library (PCL) software suite [28], using point clouds generated by a Velodyne HDL-32E sensor. In tests with falling snow, the filtered point clouds presented an improvement of over 90%, outperforming conventional outlier removal filters. Performance-wise, using this filtering technique can be quite slow, limiting the DROR utilization in real-time applications.

SOR is a denoising method that, similarly to ROR, removes the outlier points based on neighbor information (Fig. 1c). However, instead of using a fixed radius and a minimum threshold for the number of neighbors, first it calculates the average distance ***R1*** of each point to its neighbors, defined as "k-nearest neighbor", rejecting the points whose distance is higher than ***R2***, i.e., the average value plus the standard deviation. Despite improving ROR in detecting outlier points, SOR severely increases the computation overhead. For this reason, Balta *et al.* have proposed the FCSOR method [29]. Before calculating the distance to neighbors, FCSOR performs a sub-sampling of the point cloud with a VG filter step (Fig. 1a). However, and despite decreasing the computational complexity due to the reduction of the number of points, it still does not fit the real-time requirements, and the success rate in detecting outlier points slightly decreases. FCSOR was evaluated using an Intel i7-2650 4 Core (at 2.4 GHz) CPU, with 16 GB of RAM.

LIOR is a method proposed by Park *et al.* that aims at improving the speed and accuracy performance limitations of previous methods by removing the noise caused by snow or rain based on the intensity of the reflected light [30]. Noise points usually present a lower intensity value when compared with neighbors at the same distance. Thus, every point below a defined threshold value is classified as an outlier. To reduce the false positive ratio, a second step is applied to each outlier, which can be turned into an inlier if several neighbors (defined by a threshold) are detected within a specified distance. The working principle behind LIOR is based on the ROR algorithm (depicted by Fig. 1b) with the addition of the point intensity information. When comparing LIOR with the previous filtering methods, it can achieve filtering speeds up to 12x faster than SOR, and 8x faster than DROR. However, real-time filtering in high-speed vehicles is only possible if the method is applied only to certain regions of interest (ROI) rather than the full point cloud. Regarding the accuracy, the noise points can be filtered with the same efficiency as the DROR method. In their evaluation, LIOR claims to achieve a false positive ratio of 1%, while DROR reached almost 50% of points wrongly classified as outliers. LIOR was deployed and tested on an Intel Core i9-9900KF CPU (at 3.60 GHz), with 32 GB of RAM.

Dynamic low-Intensity Outlier Removal (DIOR) is a novel approach, proposed in this article, for weather denoising based on existing outlier removal techniques. DIOR follows the working principle of LIOR, which classifies outlier points based on their intensity values within a constant search radius ***R1*** (as used by ROR). However, instead of using the ROR principle, DIOR follows the DROR strategy, which classifies outlier points based on a search radius ***R1*** that dynamically increments as the distance to the objects increases. The overall workflow of DIOR is described by the pseudocode depicted in Algorithm 1. Firstly, for each point p in the point cloud, the algorithm verifies if its intensity value I_p is above the defined threshold I_{thr} to classify the point as an inlier. If this condition fails, a second step is performed by the algorithm to count the number of neighbors within the search radius ***R1***. Because ***R1*** dynamically changes as the distance to the sensor increases, before counting the number of neighbors the algorithm verifies the point's distance r_p to the sensor with Equation 2. If r_p is below the minimum search radius, ***R1*** takes the minimum search radius value SR_{min}, otherwise it defines ***R1*** using Equation 1. This Equation calculates the dynamic search radius as follows: β is a tuning parameter used to compensate the point spacing increase resulted from surfaces that are not perpendicular to the LiDAR beams (β should always be > 1); r_p is the point's distance (calculated with Equation 2); and α is the sensor's horizontal angular resolution. Next, if the number of neighbors inside ***R1*** is above the minimum threshold, the point p is classified as an inlier, otherwise, an outlier.

$$SR_p = \beta \times (r_p \times \alpha) \tag{1}$$

$$r_p = \sqrt{x_p^2 + y_p^2} \tag{2}$$

Algorithm 1 DIOR Pseudocode

```
1: for $p \in P$ do
2:     if $\mathbf{I}_p > \mathbf{I}_{thr}$ then
3:         $Inliers \leftarrow p$
4:     else
5:         if $r_p < SR_{min}$ then
6:             $\boldsymbol{R1} \leftarrow SR_{min}$
7:         else
8:             $\boldsymbol{R1} \leftarrow SR_p$
9:         $n \leftarrow NeighborSearch(p, \boldsymbol{R1})$
10:        if $n > n_{min}$ then
11:            $Inliers \leftarrow p$
12:        else
13:            $Outliers \leftarrow p$
```

III. Weather Denoising Framework

For deploying and testing weather denoising techniques, we have developed a weather denoising framework called ALFA devoted to enhance the real-time pre-processing of a 3D LiDAR point cloud. The framework is assisted by a reconfigurable hardware platform that enables the deployment of weather denoising methods in dedicated accelerators.

A. System Architecture

Fig. 2 depicts the ALFA architecture that, following a hardware-software co-design approach, enables the fast deployment and evaluation of state-of-the-art weather denoising methods both in hardware and software. Regarding the

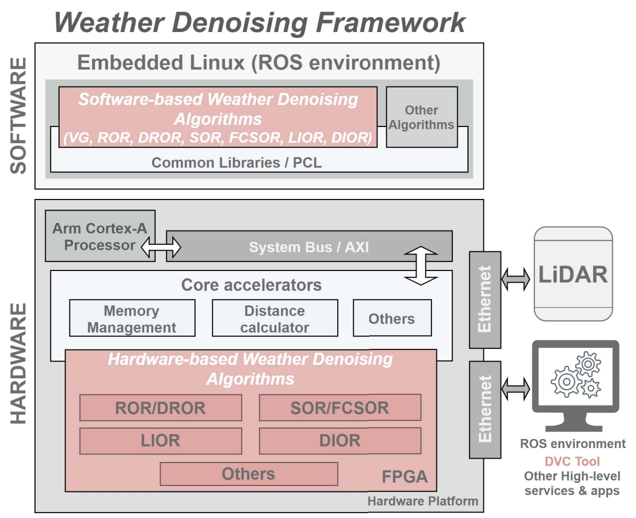


Fig. 2. ALFA architecture.

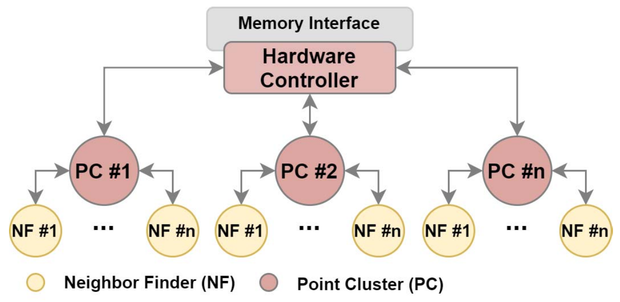


Fig. 3. ALFA hardware implementation.

software layer, ALFA features the Robot Operating System (ROS) environment on top of a minimalist embedded Linux, providing different levels of abstraction for high-level applications, e.g., the ALFA-DVC tool. This tool provides several features such as debug, a real-time point cloud visualizer, platform setup, and algorithm's configurations. The framework supports a collection of software-based denoising methods (assisted by the PCL library) and core-libraries that abstract and interface the weather denoising accelerators.

Regarding the hardware platform, ALFA is built upon the Zynq UltraScale+ XCZU7EV-2FFVC1156 MPSoC (available in the ZCU104 Evaluation Kit), enabling the design of embedded applications such as Advanced Driver Assisted Systems (ADAS) with support of video codecs and the most common peripherals and interfaces for embedded vision solutions. The 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 field-programmable gate array (FPGA) technology, and 2GB of DDR4 memory.

The communication is handled by the advanced extensible interface (AXI)-4 system bus, where the AXI4-full is used for high-performance memory accesses and the AXI4-stream for high-speed point cloud data streaming. The hardware features encompass several core blocks such as: a hardware interface for the software libraries; a memory management module; and distance calculator units to be used by the hardware weather denoising accelerator. The hardware platform also provides two Ethernet ports to respectively connect the LiDAR sensor and to make the denoised point cloud available to high-level applications.

B. Software-Based Denoising Algorithms

The weather denoising framework and the ALFA-DVC tool currently support the deployment and evaluation of the following state-of-the-art denoising algorithms: VG, SOR, FCSOR, ROR, DROR, LIOR, as well as our new method, DIOR. These software-only implementations are supported by the PCL [28] software, which provides a set of modules for point cloud processing, e.g., filtering, feature estimation, surface reconstruction, registration, model fitting, and segmentation. While VG, ROR, and SOR have a direct implementation in the PCL software, for deploying the FCSOR, DROR, LIOR, and DIOR methods, we have followed each algorithm's description and resorted to several PCL libraries.

C. Hardware-Based Denoising Algorithms

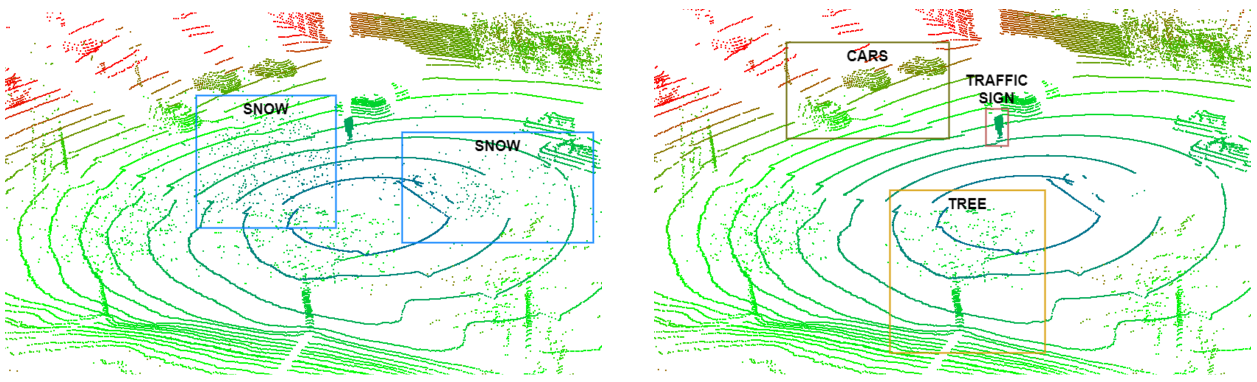
One of the key points of the ALFA framework is the ability to deploy customized hardware accelerators on the FPGA fabric. This opens the possibility to improve the performance of the supported software version of weather denoising algorithms. Their generic hardware implementation, depicted in Fig. 3, is composed of four main components: (1) the hardware controller; (2) a memory interface; (3) several Point Cluster (PC) blocks; and (4) Neighbor Finder (NF) units. Each PC block deployment requires at least 2 NFs, requiring each of them 1 Distance Calculator module available from the ALFA core accelerators (Fig. 2). The PC and NF blocks are used to parallelize the algorithm's execution in finding neighbors to classify a given point as inlier or outlier, decreasing the required time to process a point cloud frame.

The Memory Interface module is responsible for interfacing the memory where the points are stored and the hardware controller. It also provides control flow mechanisms (stop and run) to synchronize its execution with the hardware controller.

Hardware Controller is the main block of the denoising accelerator and it is responsible to: (1) read data points from memory for validation; (2) store the read points into a local buffer and allocate them into each PC and NF unit; (3) send points to the PCs for comparison; (4) check and control the output of all PCs; and (5) tag and store the points classified as outliers from each PC.

A PC module is used to store the current point under validation, i.e., to be classified as an outlier or inlier. The classification is based on the number of neighbor points that lie inside the search radius $\boldsymbol{R1}$. If the number of neighbors found is below the defined neighbor threshold, the point under validation is classified as an outlier, otherwise, the point is tagged as an inlier.

The NF module is used by the PC to calculate the distance between two given points. The output is then used by the PC to classify the point under validation considering the neighbor threshold and the search radius $\boldsymbol{R1}$.



(a) Corrupted Point Cloud data. (b) Denoised Point Cloud data.

Fig. 4. Point Cloud output before and after applying the DIOR algorithm.

D. ALFA-DVC Tool

The ALFA-DVC tool is a high-level and cross-platform QT application that runs on a desktop system. It enables the real-time visualization of point clouds (Fig. 4), allows the deployment and evaluation of software-based algorithms directly on top of a point cloud, and provides the debug and configuration of weather denoising methods. When the data points are collected in ideal weather conditions, the ALFA-DVC can also emulate and generate different weather noise applied to specific regions of the point cloud through a box system. Such box system allows to label and remove noise points, as well as to analyze the algorithm through several performance and accuracy metrics.

Point cloud data can also be loaded into the ALFA-DVC tool using Point Cloud Data (.pcd) and Polygon File Format (.ply) files stored in the file system or through the available ROS interface. This interface can provide point clouds directly received from a LiDAR sensor connected to the ALFA hardware platform or from a ROS topic running in a local ROS node. The tool also enables screenshots saving of the point cloud currently being played, as well as to publish the noised/denoised point cloud on a new ROS topic.

IV. Evaluation: Software-Based Algorithms

All software runs on top of an embedded Linux (4.19.0-xilinx-v2019.2) with the PCL library (version 1.8.1-r0) and a ROS environment (Ros1 melodic distribution). The software is supported by the ALFA hardware platform, previously described in Section III-A, with the CPU running at a clock speed of 1.2 GHz and the DDR4 memory running at 535 MHz. The FPGA was not used during the evaluation of the software-based algorithms. The tested point cloud is part of a public dataset and it was retrieved with a Velodyne Puck (VLP-16) in a real-world scenario during an intense snow storm [31]. For the evaluation, we have used the full point cloud, which contains, on average, around 17098 points per frame. The denoising algorithms were evaluated with the filter parameters depicted in Table I in the following metrics: (i) points removed (PR), Equation 3; (ii) true positives (TP), Equation 4; (iii) false positives (FP), Equation 5; (iv) false negatives (FN), Equation 6; (v) frame processing time (FPT); and (vi) frames per second (FPS). The gathered results are summarized in Table II. Regarding the DROR, LIOR, and DIOR algorithms, they were executed in a multi-thread configuration (four threads).

TABLE I
Filter Parameters

Algorithm	*Parameter*	*Value*
SOR	Number of k neighbors	4
	Standard deviation multiplier	0.9
ROR	Searching radius	0.5
	Min. number of neighbors	5
DROR	Min. searching radius	0.1
	Min. number of neighbors	30
	Angular resolution	0.3
	Radius multiplier	0.9
LIOR	Searching radius	0.5
	Min. number of neighbors	5
	Intensity threshold	4
DIOR	Min. searching radius	0.1
	Angular resolution	0.3
	Radius multiplier	0.9
	Min. number of neighbors	30
	Intensity threshold	4

TABLE II
Evaluation of Software-Only Denoising Algorithms

Algorithm	*PR*	*TP*	*FP*	*FN*	*FPT (ms)*	*FPS*
Voxel Grid	28.8%	12%	29%	87%	6	166
ROR	19%	75%	18%	24%	180	5.50
SOR	7%	56%	5%	44%	175	5.71
DROR	2%	87%	1%	13%	1193	0.83
FCSOR	34%	59%	34%	40%	117	8.5
LIOR	11%	74%	10%	25%	109	9.17
DIOR	2%	85%	0%	14%	442	2.26

$$PR = Input\ Points - Output\ Points \tag{3}$$

$$TP = \frac{Filtered\ Noise\ Points\ In\ Noise\ Areas}{Total\ Labeled\ Noise\ Points} \tag{4}$$

$$FP = \frac{Filtered\ Noise\ Points\ In\ Non\ Noise\ Areas}{Total\ Labeled\ Non\ Noise\ Points} \tag{5}$$

$$FN = \frac{Unfiltered\ Noise\ Points\ In\ Noise\ Areas}{Total\ Labeled\ Noise\ Points} \tag{6}$$

The VG algorithm outperforms the remaining methods in terms of the required processing time, i.e., it executes faster. However, it does not provide enough accuracy in finding noise, removing around 28% of the points with a false negative rate of nearly 87%. Regarding DROR, it gives better results in all metrics when compared with ROR, but it requires nearly 6.6x more processing time for denoising one frame. Concerning SOR and FCSOR, they provide a TP rate between 55% and 60% with a high FN rate of nearly 40%. Respecting LIOR, which uses ROR principles, it can provide good accuracy results while processing point cloud frames faster than the other methods. Lastly, with DIOR, which proposes the same principle of LIOR combined with DROR, it is possible to achieve great results in terms of PR (2%), FP (0%), and FN (14%), but at the cost of a slightly high FPT. Using the ALFA framework, it requires a higher processing time to filter in software one point cloud frame, which is mainly caused by the limited resources in the hardware platform, such as the PS and DDR memory speed. However, because the framework provides acceleration capabilities, we will focus on improving the processing time of the methods that provide the best TP and FP rate, i.e., DROR, LIOR, and DIOR, which we consider the most important metric in detecting noise while removing only the necessary noise points from the point cloud. These algorithms will be offloaded to dedicated hardware accelerators deployed in the FPGA. Although the other algorithms could also be accelerated, they would still keep their undesired TP, FP, and FN rate.

V. Evaluation: Hardware-Based Algorithms

The evaluation of the hardware-based denoising algorithms includes the same setup used in the software version regarding the software and hardware platform, as well as the filter parameters for the selected denoising methods. However, the hardware deployment resorts to the FPGA technology available in the hardware platform. The gathered results include the performance assessment of each algorithm for different combinations of PCs, the hardware resources required to deploy the solution in the available FPGA fabric, and the required source lines of code (SLoC) to write the different hardware modules supported by the framework.

A. Performance Evaluation

Performance-wise, the number of PC and NF dictates the required time to process one point cloud frame. In our evaluation, we have set the number of NF to 2, which is mainly related to the memory interface bandwidth used by the AXI bus, which allows fetching a total number of 2 points (16-bit representation) on each memory access. We have evaluated the trade-off between the number of PCs and the achieved FPT, while analyzing other metrics' outputs. Table III summarizes the gathered results of the performance metrics for processing the same point cloud (17098 points per frame) used in the software-only evaluation, while varying the number of PCs. Regarding the PR, TP, FP, and FN rates, the obtained values are identical to the software-only results depicted in Table II, while it is still visible that our approach provides better performance results than LIOR. Such results prove the correctness of the hardware implementation of the selected algorithms since the hardware deployment only accelerates the point cloud frame processing time without changing the algorithm's behavior.

TABLE III
Evaluation of Hardware-Based Denoising Algorithms for Different Values of PCs

Algorithm	*PCs*	*PR*	*TP*	*FP*	*FN*	*FPT (ms)*	*FPS*
DROR	2	5.8%	95%	3%	5.4%	689	1.45
	4	5.3%	95%	3%	5.5%	350	2.8
	8	5.6%	93.2%	3.4%	6.8%	170	5.89
	16	5.35%	92%	3%	7%	84	11.9
	32	4%	87%	3%	13%	40	27.77
LIOR	2	11%	73%	9%	26%	631	1.58
	4	10.65%	72.05%	8.9%	28%	308	5.31
	8	9%	73%	8.9%	30%	143	6.9
	16	9%	71%	7.6%	30%	56	19.6
	32	9%	68.2%	6.9%	34%	30	33.3
DIOR	2	5%	93%	3%	5.8%	476	2.1
	4	5%	93%	3%	5.6%	244	4.1
	8	5%	93%	3%	7%	124	8.1
	16	4.35%	91%	2.5%	7%	56	19.6
	32	4%	88%	2.8%	13%	30	33.3

TABLE IV
Hardware Resources Utilization

	FPGA Resources				
PCs	LUTs	LUTRAMs	FFs	BRAMs	DSP
2	7.79%	3.30%	4.05%	83.3%	0.92%
4	9.01%	3.30%	4.18%	82.2%	1.85%
8	11.96%	3.30%	4.39%	82.2%	3.7%
16	19.09%	3.30%	5.09%	82.2%	7.4%
32	39.30%	3.30%	5.68%	83.3%	14.81%

Regarding the processing time, and because we have deployed the algorithms with parallelization capabilities, increasing the number of PCs has an observable impact on the time required to process one point cloud frame. When the system was only deployed with 2 PCs, the processing time for DROR and DIOR is almost identical with their software implementation, while for the LIOR it results in a significant time increase from 109 ms to 631 ms. When the number of PCs is higher than 2, the results are always better than the software-only implementation, achieving an FPT of 40 ms for DROR, and 30 ms for LIOR and DIOR when the number of PCs is 32. This results in a frame rate of 27.7 FPS for DROR, and 33.3 FPS both for LIOR and DIOR.

B. Hardware Resources

The performance gain comes at the cost of FPGA hardware resources, which increases with the increase of the number of PC blocks. Table IV summarizes the trade-off between the hardware resources and the number of PCs (varying from 2 to 32). The three algorithms supported by the hardware version of the ALFA framework are all deployed inside a PC module, which can be activated on-the-fly by the ALFA-DVC tool when required. Deploying the hardware with 32 PCs requires nearly 39% of the available lookup tables (LUTs), 3.3% of lookup table random access memories (LUTRAMs), 5.68% of Flip-Flops (FFs), 83.3% of the available block RAM (BRAM), and 14.81% of digital signal processor (DSP) blocks. However, considering the frame rate achieved by the three algorithms in processing the selected point cloud, a number of 16 PCs can already provide real-time capabilities to sensors with an output of 10Hz at the cost of a few hardware resources.

TABLE V
SLoC FOR EACH HARDWARE AND SOFTWARE MODULE OF THE ALFA-PD FRAMEWORK

Module	**SLoC**	
	hardware (verilog)	**software (C)**
Memory Interface	234	—
Hardware Controller	160	—
Point Cluster (each)	128	—
Neighbor Finder (each)	26	—
Distance Calculator (each)	30	—
VG	—	4
ROR	—	4
SOR	—	5
DROR	—	33
FCSOR	—	4
LIOR	—	27
DIOR	—	37
Total	597	190

C. Source Lines of Code (SLoC)

Table V summarizes the SLoC required to implement the framework regarding the software- and hardware-based denoising algorithms, without considering the ALFA-DVC. This evaluation aims at providing an estimation of the engineering effort required for developing weather denoising algorithms. The hardware implementation of the ALFA framework is divided into four main modules, as previously explained in Section III-C. The required code for a PC or an NF is displayed for a single instance of each module. Regarding the software implementation, it is important to mention that VG, ROR, SOR, and FCSOR required just a few lines of SLoC since they directly use the PCL software library, while for the other algorithms, the implementation was still simple and did not require much code writing.

D. Closing Discussion

Comparing DROR and LIOR algorithms with our implementation (DIOR) and using the ALFA platform, the obtained results show that DIOR can achieve better performance ratios due to the combination of LIOR and DROR methods. Moreover, for the point cloud with 17098 points, DIOR achieved better frame processing times with an FPS rate up to around 33 FPS. Regarding the hardware platform, and because our processing unit provides less resources when compared with the original setups used by DROR and LIOR, the ALFA framework offers the possibility to deploy hardware accelerators to achieve better performance rates at the cost of FPGA hardware resources, even when using an embedded configuration.

However, newer LiDAR sensors can provide denser point clouds with millions of points to be processed. Despite increasing the accuracy of the perception system, they require more powerful resources to process the point cloud data, ideally in real-time. In the evaluation of the LIOR algorithm, authors have used an Ouster OS-1 LiDAR with a point cloud output of around 60k points per frame. According to their results, and before applying optimizations, the processing rate performance for processing the full sensor's data was 0.16 FPS for DROR and 1.32 FPS for LIOR. To increase the performance of their filter, authors have optimized the denoising algorithm by applying a cropbox to different ROI to reduce the number of points to be processed: Cropbox A (full data); Cropbox B (forward data); and Cropbox C (only road data). By reducing the number of points, the achieved performance speed was 9.31 FPS for the Cropbox B and 10.0 FPS for the Cropbox C. This way, authors could claim a frame rate that can cope with the real-time processing of the output of a LiDAR sensor which is commonly 10Hz.

TABLE VI
HARDWARE-BASED ALGORITHMS OPTIMIZATION

	Cropbox A (59395 points)			Cropbox B (27734 points)			Cropbox C (25456 points)		
	DROR	LIOR	DIOR	DROR	LIOR	DIOR	DROR	LIOR	DIOR
FPT	270	76	77	78	31	31	67	28	28
FPS	3.7	13.01	12.98	12.82	32	32	15	35.6	35.7

For a fair comparison between the frame rate provided by the ALFA platform and LIOR's hardware setup, which includes more processing capabilities, we have processed the point cloud from a Velodyne VLP-32C sensor, which can also output around 60k points per frame. Table VI summarizes the obtained results when applying the same strategy of including a cropbox over different ROI. Processing the full point cloud (Cropbox A) results in a frame rate of around 13 FPS, both for LIOR and DIOR, which copes with the requirement of processing a sensor's output of 10 Hz. Applying the Cropbox B and C reduces the number of points to be processed to 27734 and 25456, respectively. This results in a processing frame rate of around 32 FPS for the Cropbox B and 35 FPS for Cropbox C, showing that the ALFA hardware is able to achieve real-time capabilities even for sensors output with higher rates when ROI are applied.

VI. CONCLUSION

This article presents a new method for point cloud weather denoising called DIOR, which combines two state-of-the-art algorithms, DROR and LIOR. DIOR provides better point filtering ratios while improving the frame processing time to nearly 33 FPS when resorting to the FPGA fabric. All algorithms are supported by a weather denoising framework called ALFA that enables the implementation of different denoising methods, which can be both deployed in software and hardware. The framework also includes the ALFA-DVC tool, which enables the easy deployment and configuration of different weather denoising approaches, provides software testing and debug, and allows for point cloud visualization and integration with a ROS environment. The embedded nature of the framework shows how state-of-the-art methods can benefit from hardware acceleration to significantly improve the processing time of point cloud frames, while keeping the desired rates of the accuracy metrics used, such as the number of PR, TP, FP, and FN ratios.

In the near future, we aim at giving support to other weather denoising methods that could be accelerated in hardware. Moreover, and since the framework provides enough flexibility for supporting other tasks in hardware, we aim at expanding its functionalities to support important features such as ground segmentation and point cloud compression strategies.

References

[1] J. Guerrero-Ibáñez, S. Zeadally, and J. Contreras-Castillo, "Sensor technologies for intelligent transportation systems," *Sensors*, vol. 18, no. 4, p. 1212, Apr. 2018.

[2] E. Marti, M. A. de Miguel, F. Garcia, and J. Perez, "A review of sensor technologies for perception in automated driving," *IEEE Intell. Transp. Syst. Mag.*, vol. 11, no. 4, pp. 94–108, Winter 2019.

[3] B. S. Jahromi, T. Tulabandhula, and S. Cetin, "Real-time hybrid multi-sensor fusion framework for perception in autonomous vehicles," *Sensors*, vol. 19, no. 20, p. 4357, Oct. 2019.

[4] A. S. Mohammed, A. Amamou, F. K. Ayevide, S. Kelouwani, K. Agbossou, and N. Zioui, "The perception system of intelligent ground vehicles in all weather conditions: A systematic literature review," *Sensors*, vol. 20, no. 22, p. 6532, Nov. 2020.

[5] M. E. Warren, "Automotive LiDAR technology," in *Proc. Symp. VLSI Circuits*, Jun. 2019, pp. C254–C255.

[6] Y. Li and J. Ibanez-Guzman, "Lidar for autonomous driving: The principles, challenges, and trends for automotive LiDAR and perception systems," *IEEE Signal Process. Mag.*, vol. 37, no. 4, pp. 50–61, Jul. 2020.

[7] R. Roriz, J. Cabral, and T. Gomes, "Automotive LiDAR technology: A survey," *IEEE Trans. Intell. Transp. Syst.*, early access, Jun. 15, 2021, doi: 10.1109/TITS.2021.3086804.

[8] T. Raj, F. H. Hashim, A. B. Huddin, M. F. Ibrahim, and A. Hussain, "A survey on LiDAR scanning mechanisms," *Electronics*, vol. 9, no. 5, p. 741, Apr. 2020.

[9] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3D object detection methods for autonomous driving applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3782–3795, Oct. 2019.

[10] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 770–779.

[11] J. Wu, H. Xu, Y. Tian, R. Pi, and R. Yue, "Vehicle detection under adverse weather from roadside LiDAR data," *Sensors*, vol. 20, no. 12, p. 3433, Jun. 2020.

[12] X. Peng and J. Shan, "Detection and tracking of pedestrians using Doppler LiDAR," *Remote Sens.*, vol. 13, no. 15, p. 2952, Jul. 2021.

[13] W. Huang *et al.*, "A fast point cloud ground segmentation approach based on coarse-to-fine Markov random field," *IEEE Trans. Intell. Transp. Syst.*, early access, Apr. 21, 2021, doi: 10.1109/TITS.2021.3073151.

[14] R. Karlsson, D. R. Wong, K. Kawabata, S. Thompson, and N. Sakai, "Probabilistic rainfall estimation from automotive LiDAR," 2021, *arXiv:2104.11467*.

[15] M. Jokela, M. Kutila, and P. Pyykönen, "Testing and validation of automotive point-cloud sensors in adverse weather conditions," *Appl. Sci.*, vol. 9, no. 11, p. 2341, Jun. 2019.

[16] P. H. Chan, G. Dhadyalla, and V. Donzella, "A framework to analyze noise factors of automotive perception sensors," *IEEE Sensors Lett.*, vol. 4, no. 6, pp. 1–4, Jun. 2020.

[17] J. R. Vargas Rivero, T. Gerbich, V. Teiluf, B. Buschardt, and J. Chen, "Weather classification using an automotive LiDAR sensor based on detections on asphalt and atmosphere," *Sensors*, vol. 20, no. 15, p. 4306, Aug. 2020.

[18] R. Heinzler, P. Schindler, J. Seekircher, W. Ritter, and W. Stork, "Weather influence and classification with automotive LiDAR sensors," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2019, pp. 1527–1534.

[19] R. Heinzler, F. Piewak, P. Schindler, and W. Stork, "CNN-based LiDAR point cloud de-noising in adverse weather," *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2514–2521, Apr. 2020.

[20] T.-H. Sang, S. Tsai, and T. Yu, "Mitigating effects of uniform fog on SPAD LiDAR," *IEEE Sensors Lett.*, vol. 4, no. 9, pp. 1–4, Sep. 2020.

[21] T. Yang, Y. Li, Y. Ruichek, and Z. Yan, "LaNoising: A data-driven approach for 903 nm ToF LiDAR performance modeling under fog," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 10084–10091.

[22] Y. Li, P. Duthon, M. Colomb, and J. Ibanez-Guzman, "What happens for a ToF LiDAR in fog?" *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 11, pp. 6670–6681, Nov. 2021.

[23] C. Goodin, D. Carruth, M. Doude, and C. Hudson, "Predicting the influence of rain on LiDAR in ADAS," *Electronics*, vol. 8, no. 1, p. 89, Jan. 2019.

[24] S. Hasirlioglu and A. Riener, "A general approach for simulating rain effects on sensor data in real and virtual environments," *IEEE Trans. Intell. Vehicles*, vol. 5, no. 3, pp. 426–438, Sep. 2020.

[25] M. Byeon and S. W. Yoon, "Analysis of automotive LiDAR sensor model considering scattering effects in regional rain environments," *IEEE Access*, vol. 8, pp. 102669–102679, 2020.

[26] J. P. Espineira, J. Robinson, J. Groenewald, P. H. Chan, and V. Donzella, "Realistic LiDAR with noise model for real-time testing of automated vehicles in a virtual environment," *IEEE Sensors J.*, vol. 21, no. 8, pp. 9919–9926, Apr. 2021.

[27] N. Charron, S. Phillips, and S. L. Waslander, "De-noising of LiDAR point clouds corrupted by snowfall," in *Proc. 15th Conf. Comput. Robot Vis. (CRV)*, May 2018, pp. 254–261.

[28] R. B. Rusu and S. Cousins, "3D is here: Point cloud library (PCL)," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 2011, pp. 1–4.

[29] H. Balta, J. Velagic, W. Bosschaerts, G. De Cubber, and B. Siciliano, "Fast statistical outlier removal based method for large 3D point clouds of outdoor environments," *IFAC-PapersOnLine*, vol. 51, no. 22, pp. 348–353, 2018.

[30] J. I. Park, J. Park, and K. S. Kim, "Fast and accurate desnowing algorithm for LiDAR point clouds," *IEEE Access*, vol. 8, pp. 160202–160212, 2020.

[31] (2018). *Richard Kelley*. [Online]. Available: https://richardkelley.io/data

Ricardo Roriz received the master's degree in industrial electronics and computers engineering from the University of Minho, Portugal. He is currently pursuing the Ph.D. degree in sensors and instrumentation systems for the automotive industry. He is an Active Research Fellow at the ALGORITMI Research Center, Embedded Systems Research Group. His research interests include embedded systems design and robotics, with particular focus on FPGA-based systems.

André Campos is currently pursuing the master's degree in industrial electronics and computers engineering with the University of Minho. His research interests include embedded systems design, automotive technology, computer vision systems, and FPGA-assisted acceleration solutions.

Sandro Pinto received the Ph.D. degree in electronics and computer engineering. He is a Research Scientist and an Invited Professor at the University of Minho, Portugal. He has a deep academic background and several years of industry collaboration focusing on operating systems, virtualization, and security for embedded; cyber physical; and IoT-based systems. He is also a skilled presenter with speaking experience in several academic and industrial conferences. He has published several scientific papers in top-tier conferences/journals.

Tiago Gomes received the master's degree in telecommunications engineering and the Ph.D. degree in electronics and computers engineering from the University of Minho, Portugal. He is a Research Scientist and an Invited Professor with the University of Minho. His current research interest includes embedded systems hardware/software co-design for resource constrained wireless Internet of Things low-end devices.