applied sciences

MDPI

Article

A System Architecture of a Fusion System for Multiple LiDARs Image Processing

Minwoo Jung [1], Dae-Young Kim [2,*,†] and Seokhoon Kim [2,*,†]

[1] Center of Self-Organizing Software, Kyungpook National University, Daegu 41566, Korea
[2] Department of Computer Software Engineering, Soonchunhyang University, Asan 31538, Korea
* Correspondence: dyoung.kim@sch.ac.kr (D.-Y.K.); seokhoon@sch.ac.kr (S.K.)
† These authors contributed equally to this work.

Abstract: LiDAR sensors are extensively used in autonomous vehicles and their optimal use is a critical concern. In this paper, we propose an embedded software architecture for multiple LiDAR sensors, i.e., a fusion system that acts as an embedded system for processing data from multiple LiDAR sensors. The fusion system software comprises multiple clients and a single server. The client and server are connected through inter-process communication. Multiple clients create processes to process the data from each LiDAR sensor via a multiprocessing method. Our approach involves a scheduling method for efficient multiprocessing. The server uses multithreading to optimize the internal functions. For internal communication within the fusion system, multiple clients and a single server are connected using the socket method. In sequential processing, the response time increases in proportion to the number of connected LiDAR sensors. By contrast, in the proposed software architecture, the response time decreases in inverse proportion to the number of LiDAR sensors. As LiDAR sensors become increasingly popular in the field of autonomous driving, the results of this study can be expected to make a substantial contribution to technology development in this domain.

Keywords: fusion system; multiple LiDARs; data processing; embedded software

Citation: Jung, M.; Kim, D.-Y.; Kim, S. A System Architecture of a Fusion System for Multiple LiDARs Image Processing. *Appl. Sci.* **2022**, *12*, 9421. https://doi.org/10.3390/app12199421

Academic Editors: Zhaoqing Pan, Bo Peng and Jinwei Wang

Received: 18 August 2022
Accepted: 16 September 2022
Published: 20 September 2022

1. Introduction

In recent times, interest in autonomous driving has increased. Technology that recognizes the surrounding environment is the core technology of autonomous driving. Recognition technology recognizes the surrounding environment using cameras, LiDAR, and radar. Object detection and tracking are very important processes for recognizing the surrounding environment and maintaining control in autonomous driving. These technologies are directly related to the development of sensor technology.

A camera is the most popular sensor for recognizing surroundings. It is relatively easy to implement software that can detect static and dynamic obstacles within a camera. Such a camera can also provide high-resolution images. By means of these features, objects such as road signs, traffic lights, lanes, and obstacles can be recognized, and multiple objects can be recognized concurrently [1–4]. The disadvantage of a camera is that it is difficult to extract distance information from camera images. In order to overcome this disadvantage, many studies have focused on producing stereo cameras and developing algorithms to extract distance information [5–7]. Furthermore, owing to the error ratio of distance based on the unique characteristics of the camera, it is generally difficult to ensure consistent performance in varied environments [8,9]. A LiDAR sensor can replace a camera as recognition technology. LiDAR sensors emit a laser and measure distance by calculating the time it takes for the laser to reflect off an object and return. LiDAR sensors are classified into one-dimensional, two-dimensional, and three-dimensional based on the number of beams [10]. Three-dimensional LiDAR sensors are one of the most popular recognition sensors in the field of autonomous driving. These sensors have several advantages: they can

measure distances precisely, have a long range, and can be used in dark field environments. A 3D LiDAR sensor can have better performance than a camera in collecting the shape and posture of an object using a point cloud, making it better at object recognition [11]. Radar emits electromagnetic waves and receives scattered or reflected waves generated from objects. Based on the Doppler effect, using changes in the frequency, the relative speed and relative position of the detected obstacle are determined [12]. Radar is not affected by bad weather conditions and has the advantage of being independent of light intensity [13]. However, it has a disadvantage in that it is difficult to accurately detect metal objects such as road signs, guardrails, and static objects using radar.

Thus, a technology that maximizes the advantages of individual sensors through sensor fusion and improves the performance reliability of object recognition for the surrounding environment is essential. Approaches to improve the reliability of recognition technology for obstacles and surrounding environments using fusion LiDAR sensors and cameras are being extensively researched. Given that a context-awareness system needs multiple cameras, LiDAR sensors, and radar, the technology to achieve fusion between their respective data is also needed. Multiple LiDAR must be installed in the moving vehicle in order to recognize the omnidirectional environment. As autonomous vehicles require five or more LiDAR sensors, a fusion system that can control the data of multiple LiDAR sensors is needed. Such a fusion system must consider various technical elements such as synchronization among sensors, the processing time of input data, and the accuracy of object recognition.

We propose a software architecture for a fusion system that connects multiple LiDAR sensors. First, we describe a CPU scheduler to efficiently manage the processes of multiple clients that receive data from multiple LiDARs. Second, we propose a multithreading method for efficient data processing on a server. Third, we propose the idea of inter-processing communication between client and server. As mentioned above, we propose a software architecture for a fusion system that merges data from multiple LiDAR sensors. The proposed software architecture is loaded into the fusion system and the response time is measured through performance evaluation. It was confirmed that the response time of the client towards multiple LiDARs decreases, but the response time of the server increases. However, the system could be stabilized by convergence at a specific time. These results can be a basis for lightening of the fusion system.

2. Background

Research on algorithms for fusion systems using recognition sensors is being actively conducted in various fields. Schneider et al. [14] proposed a method for fusion of the raw data of a LiDAR sensor and a camera. In [15], a fusion system that works in real time is proposed for a single 3D LiDAR sensor and multiple 2D LiDAR sensors. In [16], an algorithm for collecting and recognizing data from recognition sensors is implemented using multiple LiDAR sensors. In [17], their article proposes a multiple instance multiresolution fusion (MIMRF) framework. The proposed MIMRF can fuse multiresolution and multimodal sensor outputs from multiple sensors while effectively learning from bag-level training labels. Borgmann et al. [18] implemented an algorithm to detect pedestrians in the same way as [16]. In [19], they proposed a data fusion approach to integrate low-density LiDAR data and a single optical image for a precise estimation of the tree top heights at single tree level [20]. Refs. [16,18] propose low-level fusion methods, as they directly merge raw point clouds. In [21], a detection algorithm using dual LiDAR is implemented for low-level centralized fusion [22]. In [23], a multiple feature-based superpixel-level decision fusion (MFSuDF) framework that takes full advantage of feature discriminability of different modules and superpixel structure is proposed for hyperspectral images (HSIs) and LiDAR sensor data classification. In [24], they propose a new feature fusion framework based on deep neural networks (DNNs). The proposed framework employs deep convolutional neural networks (CNNs) to effectively extract features of multi-/hyperspectral and light detection and ranging data. This research deals with applications using fusion sensors. In

order to improve their performance, they need an efficient data fusion algorithm. The data fusion algorithm is the process that integrates the data of multiple sensors to produce more consistent, accurate, and useful information than that provided by the data of an individual sensor. Data fusion approaches can be distinguished according to the level at which data are fused: fusion of raw data, fusion of features extracted from the data, and fusion of decisions derived from the data [25]. Most of the research for data fusion is focused on the high level, that is, integration of separately and independently processed sensor signals [26]. We believe that the robustness gained through independent processing in a high-level fusion setup does not have to be sacrificed if the low-level fusion architecture and the modeling process are designed properly. Although some promising initial research into low-level fusion has already been performed, raw data fusion still remains a challenging task. We consider a low-level fusion architecture that deals with received raw data from each sensor and can optimize processing delay.

2.1. Problem Definition

For data collection from multiple LiDAR sensors, the sequential data processing method has a limitation in processing time. The fusion procedure for multiple LiDAR sensors involves the collection, parsing, transformation, and formatting of LiDAR data. Processing time in the fusion system is determined by system requirements. Data processing should be completed within 33 ms in 16-channel 30 Hz LiDAR sensors. However, in the data processing method based on a single process, the processing time increases linearly according to the number of connected LiDAR sensors. Therefore, the single processing method is difficult to apply in a highly complex sensor network. Figure 1 shows the processing time of a client and a server using sequential processing. We can observe that it is impossible when the number of connected LiDAR sensors is more than two. To use n multiple LiDAR sensors where each sensor has the T_{sensor} processing time, total data processing time ($T_{process}$) for multiple LiDAR sensors in the fusion system should be as low as within the threshold, $T_{threshold}$ (i.e., 33 ms), as the service requirement.

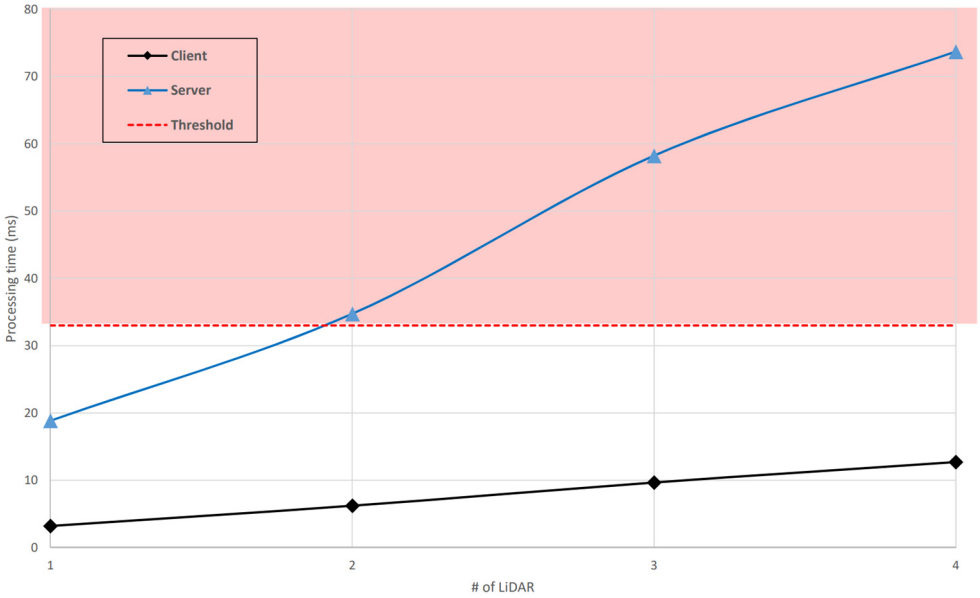

Figure 1. Response time of a fusion system using sequential processing.

Problem:

$$T_{process} \leq T_{threshold},$$
$$T_{process} = \sum_{i=1}^{n} T_{sensor}^{i}$$

2.2. Contribution

In order to overcome the limitations of sequential data processing methods, we propose a fusion system architecture for multiple LiDAR sensors that considers multiprocessing for multiple clients with multithreading within a server process. The proposed system can reduce the processing time for multiple LiDAR sensors by maximizing the resource usage of the dual core. In Figure 2, the left figure illustrates sequential data processing, in which the client receives the data of LiDAR sensors through the socket, sends it to the server after processing, and then receives the next data from the LiDAR sensors and processes them. The sequential data processing method sequentially accesses each port input from LiDAR sensors in a single process, and then collects, converts, and transmits data to the server. In this process, the processing of a LiDAR sensor must be completed to process the next data of the LiDAR sensor. This makes it difficult to ensure performance in real time in the complete system because a delay proportionate to the number of connected LiDAR sensors occurs. In Figure 2, the right side shows the processing method that can ensure concurrency through multiprocessing by creating multiple clients and sockets to receive data from multiple LIDAR sensors simultaneously. The proposed processing method can ensure performance in real time because it processes data at a fast processing time, but the efficiency varies as per the scheduling technique. We consider a scheduling method that is suitable for processing data from multiple LiDAR sensors.

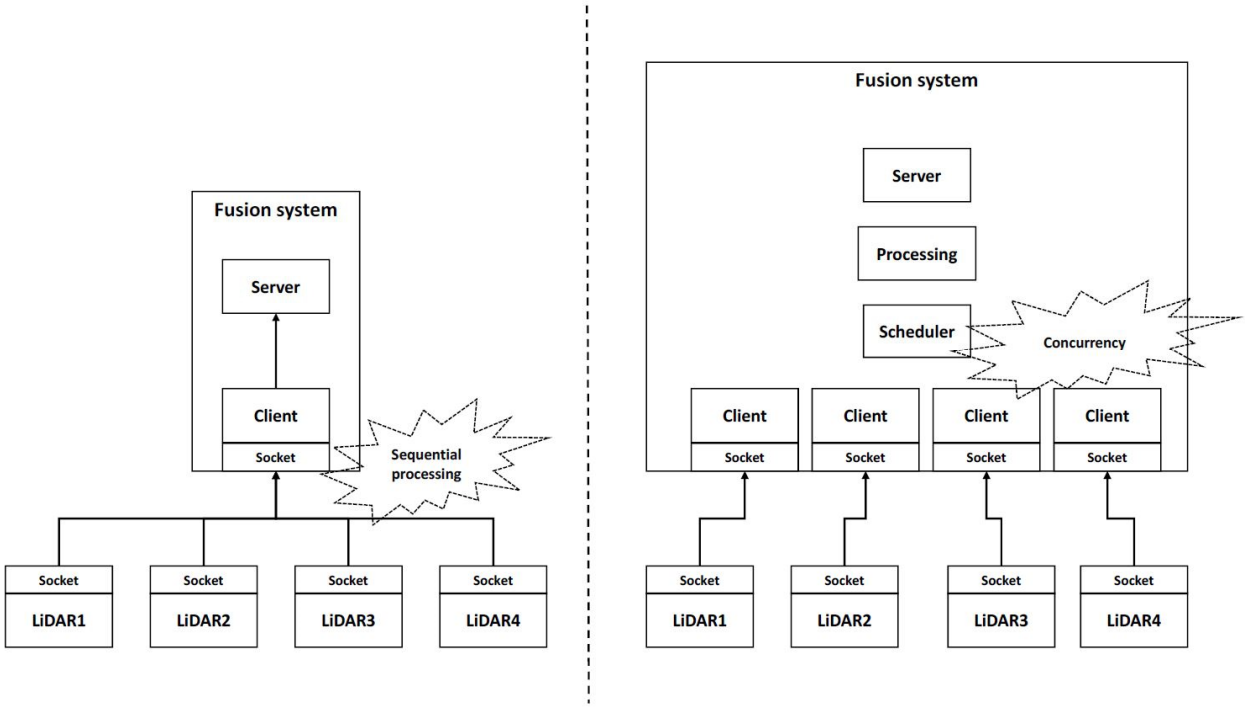

Figure 2. Sequential processing vs. multiprocessing.

3. Proposed System Architecture for Multiple LiDARs Image Processing

The data fusion system is divided into hardware and software. The hardware has communication interfaces for receiving data from LiDAR sensors, a multi-core CPU and memory for supporting multiprocessing, and communication interfaces to output the data of processing results. The operation procedures of the fusion system are as follows. The client sequentially receives data through sockets, and then the process collects, parses, and converts the data. The converted data are transmitted to the server, where the data are merged, compressed, and transmitted to the external system. Data processing involves a client process for the individual LiDAR sensor and a server process for aggregation and calculation of the point cloud. To process data input from an individual LiDAR sensor, as many clients are created as the number of sensors. The functions of the client process

are data conversion and parsing. Input data to multiple clients are transmitted to the server using Internal Processor Communication (IPC). The Figure 3 shows the overall block diagram of proposed architecture that consists of multiple clients, a server, and IPC.

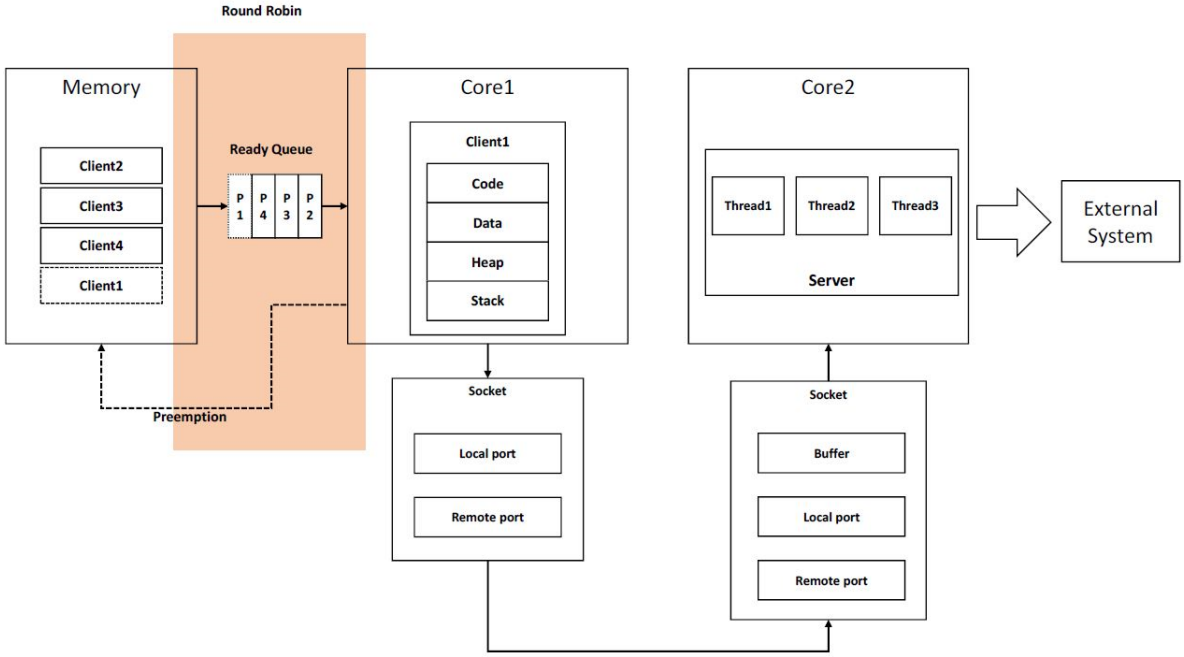

Figure 3. The proposed architecture of a fusion system.

3.1. Inter-Process Communication

To effectively implement multiprocessing, it is most important to optimize the internal communication method between processes. Communication between multiple processes involves a shared memory model and a message-passing model. In shared memory, communication between multiple processes can read and write to the shared memory area through sharing a part of the address space. When the shared memory is set, communication between multiple processes is possible without utilizing the kernel. As a result, the processing is fast and freedom of communication is ensured. However, a unique algorithm for synchronization between processes is needed without the kernel. In order to control concurrent access, an access control method such as locking or semaphores should be introduced. In this study, it is difficult to solve the synchronization problem of shared memory because there are many processes for multiple clients and a server.

Message passing can exchange data using buffering in the kernel without sharing memory between processes. It has the disadvantage of slower processing speed than shared memory, but it can be implemented in a simple manner. Message passing has a variety of models such as PIPE, Message Queue, and Socket. In this study, the socket method is used to implement communication between multiple clients and a single server. In the socket method, multiple processes can access through the abstract port provided by the operating system. In the IPC socket method, a process can communicate with another process using the port. Unlike other IPC methods, it is independent of process location and machine boundary. Communication between multiple clients and a server is conducted by UDP. In a fusion system for multiple LiDAR sensors, it is more important to transmit all the data in real time than to ensure the reliability of a packet. A UDP socket does not have a transmission buffer, so data are copied to the kernel area and deleted immediately after transmission. In this system, there is no transmission buffer in the client socket because of unidirectional communication. Figure 4 shows the inter-process communication between a client and a server through UDP protocol.

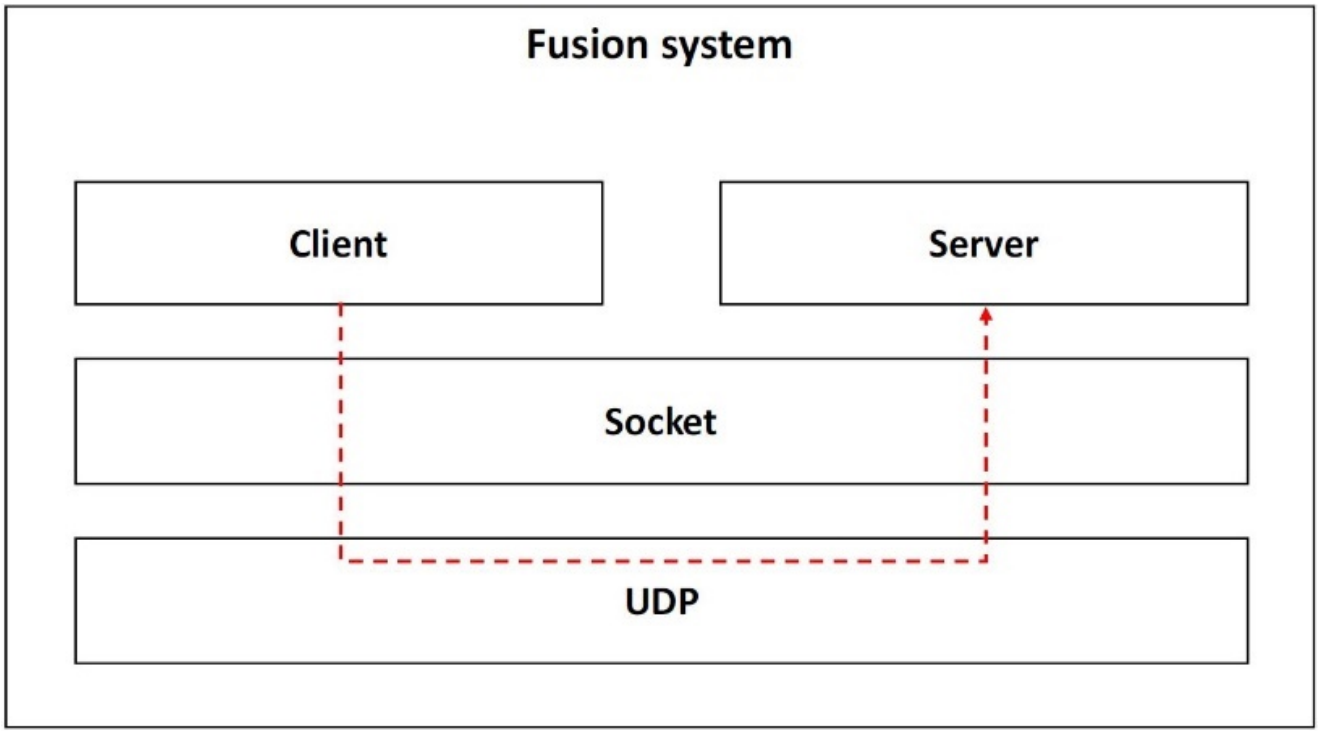

Figure 4. IPC in the fusion system.

3.2. Client Structure

The processes of a fusion system involve multiple clients and a server. A fusion system receives data from multiple LiDAR sensors through UDP communication. The client is an I/O-bound process and is deployed with the same number of LiDAR sensors. The client repeatedly loads and executes a process on the CPU through context switching. Each client includes each socket, so the processed data is transmitted to the server. As the client process frequently switches context, scheduling for processes is needed to optimize the usage of the core.

The scheduler is composed of a task scheduler and a CPU scheduler. The task scheduler is responsible for deciding the process to insert into the ready queue. The CPU scheduler considers the priority among the processes loaded in memory and decides which process should be loaded on the CPU in the order of priority. The CPU scheduler chooses a process to allocate to the CPU as per a defined scheduling algorithm. Performance metrics for scheduling are generally based on processor utilization, throughput, turnaround time, waiting time, and response time. In this study, the scheduling algorithm is implemented considering the turnaround time, waiting time, and response time. Turnaround time refers to the time wait from arrival to the first start. Waiting time refers to waiting time in the Ready Queue. Response time refers to the total time taken from arrival to completion. To optimize the scheduler, the turnaround time, waiting time, and response time need to be minimized.

Scheduling for the client processes operates by non-preemptive scheduling. In this study, a round-robin method is introduced to process the point cloud from multiple LiDAR sensors in real time. When the point cloud for multiple LiDAR sensors is input in real time, it is difficult to determine the priority based on a numerical value such as the amount of work. To have a multiple LiDAR system, a round-robin method that can be performed only within a limited processing time (called a quantum) without priority was considered. The quantum is set within the time when the next data is input from multiple LiDAR sensors. If the quantum is set to be long, it operates in a similar manner to the non-preemption method, whereas if it is set to be short, context switching occurs frequently, increasing overhead. The performance of the round-robin method depends on the quantum setting.

3.3. Server Structure

The server is a CPU-bound process, i.e., it implements algorithms having a large number of calculations. The receive buffer is set based on the size of one frame of each LiDAR sensor and the number of connected LiDAR sensors. In the server, the threshold value of the unit frame processing time of each LiDAR sensor is set based on the input period of the LiDAR sensors. The server has three functions: data aggregation, compression, and transmission. The server process operates continuously by monopolizing one core and consists of three threads. Multithreading involves two or more threads performing a task simultaneously within one process. Multithreading has the advantage of reducing the overload of system resources by sharing the memory of the process. There are operations such as object recognition by collecting the data of multiple LiDAR sensors, but we do not consider them in this study. We consider only the operation of collecting and compressing data and transmitting them to the external system. Scheduling is necessary to implement efficient multithreading. Multithreading occupies less memory space and context switching is fast because of the absence of cache updates. This system applies a multi-threading method with concurrency instead of parallelism. Scheduling is implemented in a way that considers the priority. Among the three functions, the thread collecting data has the highest priority, followed by data compression and transmission to the external system. The priority can be controlled using code assigned to the thread object. Thread scheduling is divided into two types: local scheduling, in which thread scheduling is determined by the process to which the thread belongs, and global scheduling, in which the kernel's CPU scheduler determines scheduling. In this study, there is no system call for synchronization, and multithreading is implemented based on local scheduling without context switching to maximize performance. The server operates the collection thread until data reception from all connected clients is completed, and the collection thread occupies the CPU whenever the socket receive buffer is full. Figure 5 shows the structure of multiple threads within the server process. A thread library provides the programmer an API for creating and managing threads. It provides a library entirely in user space with no kernel support and makes it possible that invoking a function in the library results in a local function call in the user space.

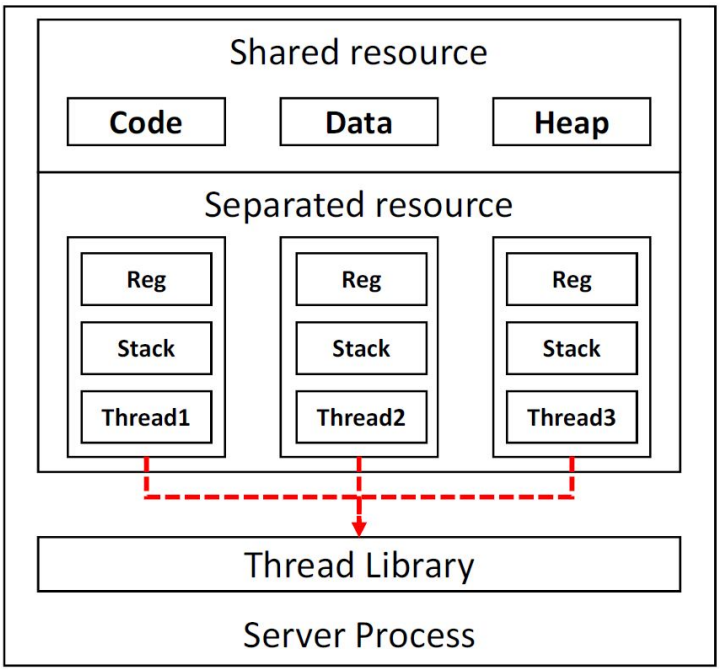

Figure 5. Multithreading in the server process.

4. Implementation and Evaluation

4.1. Experimental Environment

The fusion system is connected to up to four LiDAR sensors through Ethernet. It exports the processed data to the external system through wired or wireless communication. A high-resolution LiDAR sensor requires a lot of resources for data processing because the amount of data is very large. We consider a software architecture of the fusion system using the lightening of the algorithm in the embedded system environment. The hardware platform for the proposed system is manufactured using NXP's LS1028A dual-core general-purpose process. Five Ethernet ports are configured to connect multiple LiDAR sensors. It also includes wireless interfaces such as WiFi and LTE to communicate with external systems. The LiDAR sensors are made by Carnavicom Co., Ltd. and the sensors employ 16 channels. Each LiDAR sensor has a wide field of view which has up to 145°, 18,560 points per frame, and a sampling rate of 30 Hz. LiDAR sensors and the fusion system communicate with UDP based on port and IP. The fusion system includes functions such as collection, parsing, conversion, and formatting of data. Figure 6 shows an embedded hardware and block diagram of the fusion system for the experiment.

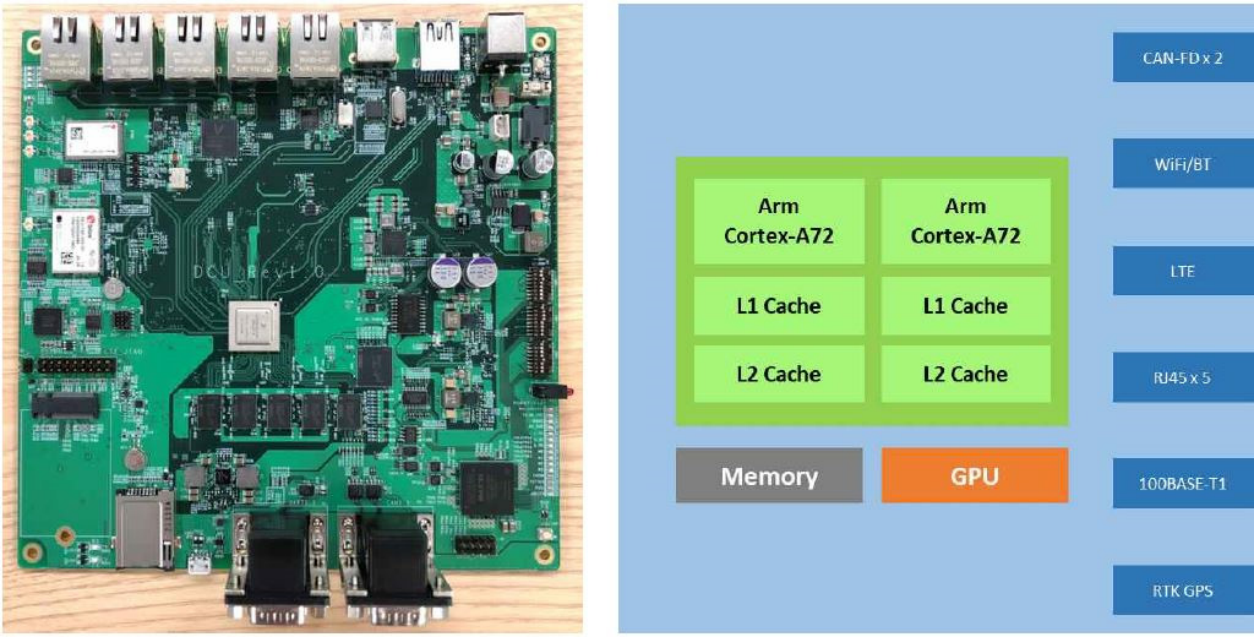

Figure 6. Hardware platform of the fusion system.

For the evaluation, the response time is measured by varying up to four LiDAR sensors connected to the fusion system. This response time refers to the process life cycle of the clients and the server in the fusion system for multiple LiDAR sensors. The response time of the client process means the sum of initialization time, waiting time, operation time, and transmission time to the server. The response time of the server process means the time taken to receive data from the clients and export the data to the external system. As many client threads are created as the number of connected LiDAR sensors. These threads are allocated to core1, and the server exclusively allocates core2. The proposed system architecture is mounted on the fusion system. We implemented GUI to evaluate the point cloud and response time of the LiDAR sensor.

In Figure 7, the top figures show the connection of LiDAR sensors and the proposed fusion system, the bottom figures show the GUI to evaluate the point cloud of the LiDAR sensors and the overall block diagram of the proposed fusion system for multiple LiDAR sensors.

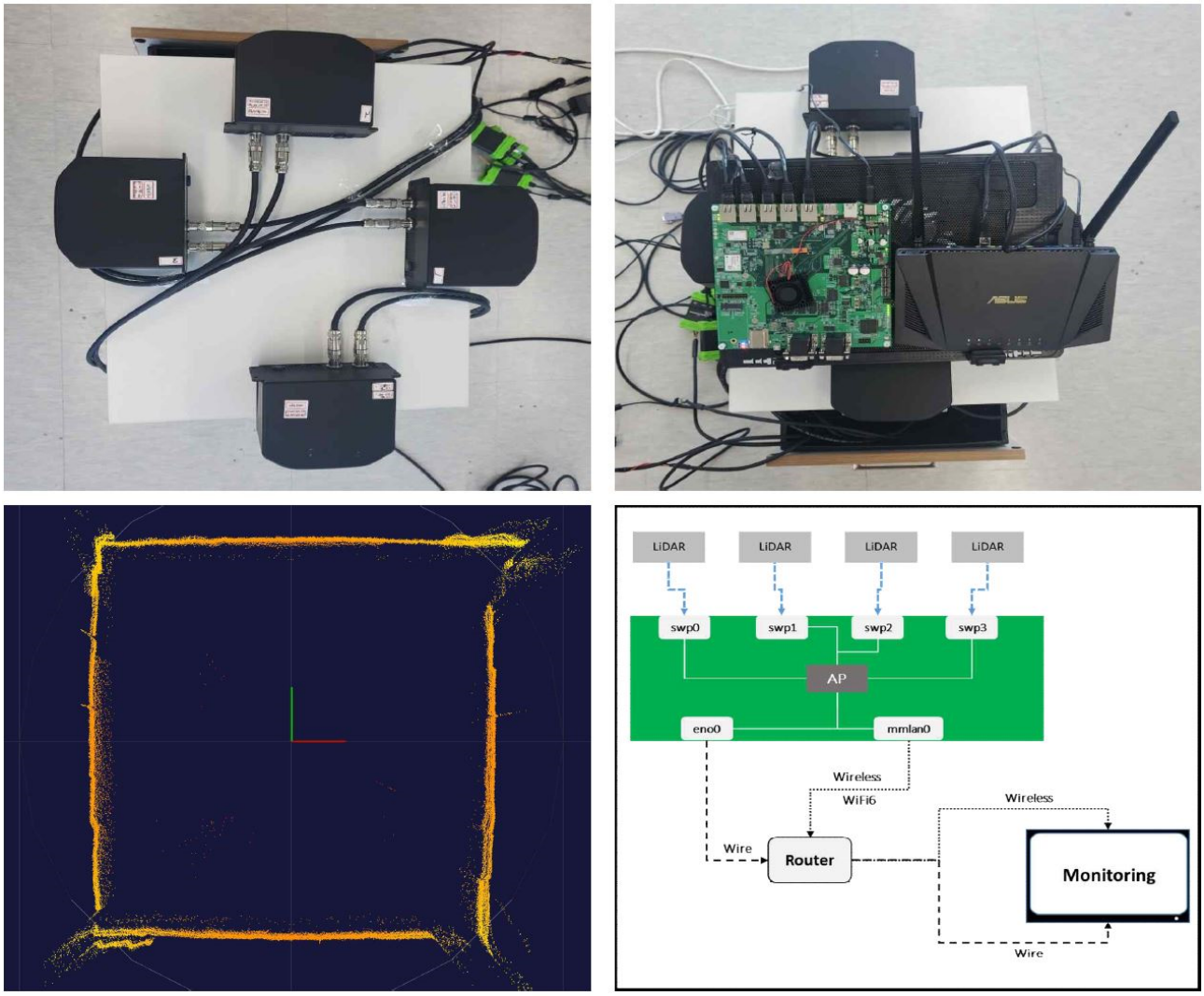

Figure 7. Experimental environment using multiple LiDARs.

4.2. Experimental Result

We measured the response time of the fusion system according to the number of connected LiDAR sensors. In the experiment, when a single LiDAR sensor is connected to the data fusion system, the client response time is approximately 20 ms. In contrast, the server response time is 4 ms.

Figure 8 represents the response time comparison between sequential processing (*Server_s*, *Client_s*) and multiprocessing (*Server_m*, *Client_m*). The red vertical lines present the differential of response time between sequential processing and multiprocessing. As the number of LiDARs is increasing, the differential of response time also has increased. Although the server response time is the same, the client response time is different. As we mentioned earlier, the sequential processing leads to linear increase in response time when the number of LiDAR sensors is increased. However, since the multiprocessing is capable of concurrent processing, the response time gradually converges as the number of LiDAR sensor increases. Therefore, as the number of connected LiDAR sensors increases, the gap of the response time between sequential processing and multiprocessing is growing.

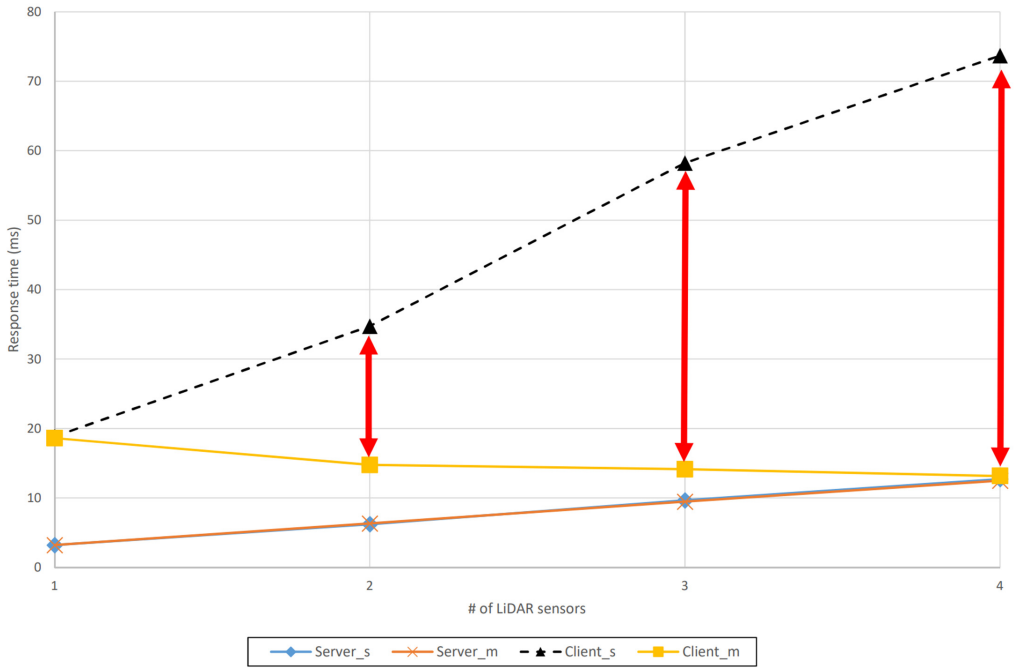

Figure 8. Response time comparison between sequential processing and multiprocessing.

Figure 9 represents the response time of the clients when varying from one to four the number of connected LiDAR sensors. As the number of connected LiDAR sensors increases, the response time of the fusion system stabilizes. It is observed that the client response times converge to approximately 13 ms as shown in Figures 8 and 9. Furthermore, in the experiment results the response time of the client decreases, but the response time of the server increases based on the amount of merged data. The reason is that inside the fusion system, multiple clients process concurrently and only one server processes the clients' data. In the fusion system, although a single server is used to process multiple LiDAR sensors' data, the response time does not exceed the processing threshold.

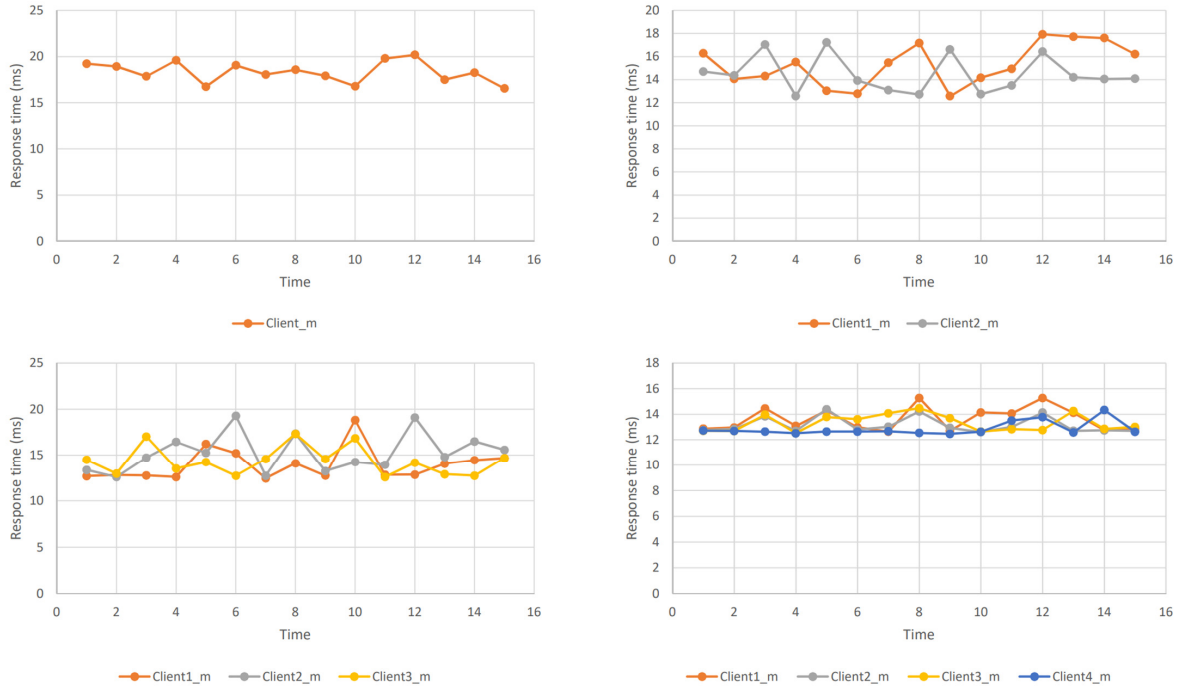

Figure 9. Response time comparison according to the number of connected LiDARs.

5. Conclusions

This paper proposes an efficient software architecture for a fusion system that processes data from multiple LiDAR sensors. Through an experiment it was confirmed that when the LiDAR sensor was processed through a client, the response time increased linearly, and therefore it is generally difficult to process multiple LiDAR sensors. Multiple clients were implemented to process data from multiple LiDAR sensors, and a round-robin scheduler was implemented to run multiprocessing. In the experiment, the response time decreased as the number of LiDAR sensors increased. It was confirmed that the response time could be shortened by preferentially processing the client whose receive buffer is full while other clients are receiving data. The server process was implemented as a priority-based multithread to improve efficiency. When the receive buffer is full, the collection thread is prioritized through an interrupt. The server process confirmed that the response time increased in proportion to the number of LiDAR sensors. In terms of the overall system, it was confirmed that the response time converges at a specific time as the number of LiDAR sensors increased. We can predict that if the point cloud increases, response time also increases because of the increasing of processing time.

The results of this study can be utilized in autonomous driving systems where the number of installed sensors is high. In the future, system optimization can be achieved through analyses of various scheduling techniques. We will also design a software architecture that applies an object recognition algorithm for use in the field of autonomous driving.

Author Contributions: Conceptualization, M.J.; methodology, M.J. and D.-Y.K.; software, M.J.; validation, M.J., D.-Y.K. and S.K.; formal analysis, M.J., D.-Y.K. and S.K.; investigation, M.J.; resources, M.J.; data curation, M.J. and D.-Y.K.; writing—original draft preparation, M.J.; writing—review and editing, D.-Y.K. and S.K.; visualization, M.J.; supervision, D.-Y.K. and S.K.; project administration, M.J., D.-Y.K. and S.K.; funding acquisition, M.J. and S.K. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (grant No. 2018R1D1A1B07041296); this work was supported by an Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (RS-2022-00167197, Development of Intelligent 5G/6G Infrastructure Technology for The Smart City); this work was funded by BK21 FOUR (Fostering Outstanding Universities for Research) (no. 5199990914048); and this work was supported by the Soonchunhyang University Research Fund.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Campbell, S.; O'Mahony, N.; Krpalcova, L.; Riordan, D.; Walsh, J.; Murphy, A.; Ryan, C. Sensor technology in autonomous vehicles: A review. In Proceedings of the 2018 29th Irish Signals and Systems Conference (ISSC), Belfast, UK, 21–22 June 2018.
2. Jahromi, B.S.; Tulabandhula, T.; Cetin, S. Real-time hybrid multi-sensor fusion framework for perception in autonomous vehicles. *Sensors* **2019**, *19*, 4357. [CrossRef] [PubMed]
3. Jung, J.; Park, M.; Cho, K.; Mun, C.; Ahn, J. Intelligent hybrid fusion algorithm with vision patterns for generation of precise digital road maps in self-driving vehicles. *KSII Trans. Internet Inf. Syst.* **2020**, *14*, 3955–3971. [CrossRef]
4. Kook, J. The design, implementation, demonstration of the architecture, service framework, and applications for a connected car. *KSII Trans. Internet Inf. Syst.* **2021**, *15*, 637–657. [CrossRef]
5. Joglekar, A.; Joshi, D.; Khemani, R.; Nair, S.; Sahare, S. Depth estimation using monocular camera. *Int. J. Comput. Sci. Inf. Technol.* **2011**, *2*, 1758–1763.
6. Bhoi, A. Monocular depth estimation: A survey. *arXiv* **2019**, arXiv:1901.09402. [CrossRef]
7. Grag, R.; Wadhwa, N.; Ansari, S.; Barron, J.T. Learning single camera depth estimation using dual-pixels. In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) 2019, Seoul, Korea, 27 October–2 November 2019. [CrossRef]

8. Cronin, C.; Conway, A.; Walsh, J. State-of-the-art review of autonomous intelligent vehicles (aiv) technologies for the automotive and manufacturing industry. In Proceedings of the 2019 30th Irish Signals and Systems Conference (ISSC), Maynooth, Ireland, 17–18 June 2019. [CrossRef]
9. Harapanahalli, S.; Mahony, N.O.; Hernandez, G.V.; Campbell, S.; Riordan, D.; Walsh, J. Autonomous navigation of mobile robots in factory environment. *Procedia Manuf.* **2019**, *38*, 1524–1531. [CrossRef]
10. Kodors, S. Point distribution as true quality of lidar point cloud. *Balt. J. Mod. Comput.* **2017**, *5*, 362–378. [CrossRef]
11. Carballo, A.; Lambert, J.; Monrroy, A.; Wong, D.; Narksri, P.; Kitsukawa, Y.; Takeuchi, E.; Kato, S.; Takeda, K. Libre: The multiple 3d lidar dataset. In Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 19 October–13 November 2020. [CrossRef]
12. Wang, Z.; Wu, Y.; Niu, Q. Multi-sensor fusion in automated driving: A survey. *IEEE Access* **2019**, *8*, 2847–2868. [CrossRef]
13. Yeong, D.J.; Barry, J.; Walsh, J. A review of multi-sensor fusion system for large heavy vehicles off road in industrial environments. In Proceedings of the 2020 31st Irish Signals and Systems Conference (ISSC), Letterkenny, Ireland, 11–12 June 2020. [CrossRef]
14. Schneider, S.; Himmelsbach, M.; Luettel, T.; Wuensche, H.-J. Fusing vision and lidar—Synchronization, correction and occlusion reasoning. In Proceedings of the 2010 IEEE Intelligent Vehicles Symposium (IV), La Jolla, CA, USA, 21–24 June 2010. [CrossRef]
15. Peterson, K.; Ziglar, J.; Rybski, P.E. Fast feature detection and stochastic parameter estimation of road shape using multiple lidar. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems 2008, Nice, France, 22–26 September 2008. [CrossRef]
16. Sualeh, M.; Kim, G.-W. Dynamic multi-lidar based multiple object detection and tracking. *Sensors* **2019**, *19*, 1474. [CrossRef] [PubMed]
17. Du, X.; Zare, A. Multiresolution multimodal sensor fusion for remote sensing data with label uncertainty. *IEEE Trans. Geosci. Remote Sens.* **2020**, *58*, 2755–2769. [CrossRef]
18. Borgmann, B.; Schatz, V.; Kieritz, H.; Scherer-Klöckling, C.; Hebel, M.; Arens, M. Data processing and recording using a versatile multi-sensor vehicle. In Proceedings of the ISPRS Annals of Photogrammetry. Remote Sensing & Spatial Information Science 2018 IV-1, Karlsruhe, Germany, 10–12 October 2018. [CrossRef]
19. Paris, C.; Bruzzone, L. A three-dimensional model-based approach to the estimation of the tree top height by fusing low-density lidar data and very high resolution optical images. *IEEE Trans. Geosci. Remote Sens.* **2015**, *53*, 467–480. [CrossRef]
20. Moon, J.; Jung, M. Geometrical properties of spilled oil on seawater detected using a lidar sensor. *J. Sens.* **2020**, *2020*, 5609168. [CrossRef] [PubMed]
21. Fortin, B.; Lherbier, R.; Noyer, J.-C. A track-before-detect approach for extended target tracking in multi-lidar systems using a low-level centralized fusion framework. In Proceedings of the 17th IEEE International Conference on Information Fusion (FUSION) 2014, Salamanca, Spain, 7–10 July 2014.
22. Jung, M.; Kim, D.-Y.; Kim, S. Efficient remote software management method based on dynamic address translation for IoT software execution platform in wireless sensor network. *Indian J. Sci. Technol.* **2016**, *9*, 1–7. [CrossRef]
23. Jia, S.; Zhan, Z.; Zhang, M.; Xu, M.; Huang, Q.; Zhou, J.; Jia, X. Multiple feature-based superpixel-level decision fusion for hyperspectral and lidar data classification. *IEEE Trans. Geosci. Remote Sens.* **2021**, *59*, 1437–1452. [CrossRef]
24. Chen, Y.; Li, C.; Ghamisi, P.; Jia, X.; Gu, Y. Deep fusion of remote sensing data for accurate classification. *IEEE Geosci. Remote Sens. Lett.* **2017**, *14*, 1253–1257. [CrossRef]
25. Jakub, S.; Anna, S.; Marina, C. Multi-sensor data fusion and parallel factor analysis reveals kinetics of wood weathering. *Talanta* **2021**, *225*, 122024. [CrossRef]
26. Rövid, A.; Remeli, V. Towards Raw Sensor Fusion in 3D Object Detection. In Proceedings of the Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, Slovakia, 24–26 January 2019. [CrossRef]