Article

Semantic Depth Data Transmission Reduction Techniques Based on Interpolated 3D Plane Reconstruction for Light-Weighted LiDAR Signal Processing Platform

Taewon Chong 1,2,†, Dongkyu Lee 3†, and Daejin Park 3,*

1 Carnavicom Co., Ltd., Incheon 21984, Korea; twchong@carnavi.com
2 Department of Physics, Hanyang University, Seoul 04763, Korea
3 School of Electronic and Electrical Engineering, Kyungpook National University, Daegu 41566, Korea; dktlee1215@knu.ac.kr
* Correspondence: boltanut@knu.ac.kr; Tel.: +82-53-950-5548
† These authors contributed equally to this work.

Abstract: In vehicles for autonomous driving, light detection and ranging (LiDAR) is one of the most used sensors, along with cameras. LiDAR sensors that produce a large amount of data in one scan make it difficult to transmit and calculate data in real-time in the vehicle’s embedded system. In this paper, we propose a platform based on semantic depth data-based data reduction and reconstruction algorithms that reduce the amount of data transmission and minimize the errors between original and restored data in a vehicle system using four LiDAR sensors. The proposed platform consists of four LiDAR sensors, an integrated processing unit (IPU) that reduces the data of the LiDAR sensors, and the main processor that reconstructs the reduced data and processes the image. In the proposed platform, the 58,000 bytes of data constituting one frame detected by the VL-AS16 LiDAR sensor were reduced by an average of 87.4% to 7295 bytes by the data reduction algorithm. In the IPU placed near the LiDAR sensor, the memory usage increased by the data reduction algorithm, but the data transmission time decreased by an average of 40.3%. The transmission time where the vehicle’s processor received one frame of data decreased from an average of 1.79 ms to 0.28 ms. Compared with the original LiDAR sensor data, the reconstructed data showed an average error of 4.39% in the region of interest (ROI). The proposed platform increased the time required for image processing in the vehicle’s main processor by an average of 6.73% but reduced the amount of data by 87.4% with a decrease in data accuracy of 4.39%.

Keywords: digital signal processing; light detection and ranging (LiDAR); embedded system; interpolation; low power system design

1. Introduction

As research on autonomous driving and, consequently, sensors used in vehicles is being actively conducted, research on sensors used in vehicles is also being conducted. In the case of autonomous vehicles using cameras, 3D object recognition research is being actively conducted. Haq et al. [1] conducted research to improve heatmap prediction using the center regression algorithm, Ul Haq et al. [2] conducted research to improve image recognition performance with a platform that performs both 2D and 3D object detection. Camera, ultrasonic and radio detection, and ranging (Radar) sensors, which are representative sensors, are installed in autonomous vehicles; however, they are used in limited areas owing to their advantages and disadvantages. The object recognition ability of an image-based camera sensor is higher than that of other sensors; however, it is significantly reduced when the field of view (FOV) and illumination are limited. In addition, it is difficult for a camera sensor to accurately measure the distance to an object. Ultrasonic sensors, which are cheaper than other sensors, have shorter detection distances. Radar sensors can
measure long distances compared to other sensors. However, the fixed object detection performance is low, and it cannot distinguish the types of the detected objects.

LiDAR sensors were introduced to compensate for the shortcomings of existing sensors. They measure the positions of objects by analyzing the light reflected from the objects after emitting light amplification by stimulated emission of radiation (laser) and control the measurement ranges using a rotating mirror. According to specifications, such as angular resolution, FOV, and the number of channels, the LiDAR sensors generate a massive amount of point clouds and transmit them to the processors of the vehicles. The range within which the LiDAR sensors can recognize objects is determined by the FOV angle. Therefore, a vehicle requires multiple LiDAR sensors to detect the entire range. As the sensors increase, the data increase, causing a data transmission bandwidth problem and processor load problem for real-time data processing in the vehicle. To reduce the processor load, preprocessing hardware was developed to preprocess the data of the LiDAR sensors using an integrated processing module. Research on compressing LiDAR data have been conducted to solve limited data bandwidths.

We propose a platform that compresses the raw data in the device around the LiDAR sensor, transmits it, and restores it in the main processor to process the data of the LiDAR sensor efficiently, which generates a large amount of data, as shown in Figure 1. Existing data compression platforms to reduce the amount of data transmission compresses the raw data regardless of the characteristics of the data produced by the LiDAR sensor, and accordingly, important features can be lost in the data during the reconstruction process. The proposed platform reduces data while preserving important object information in raw data based on the semantic depth data. From the LiDAR raw data, semantic depth data are extracted based on the amount of time change using a data compression algorithm executed in the peripheral device. The extracted data are transferred to the external main processor using the LiDAR compression protocol and restored before other algorithms, such as object recognition, are executed. The reconstruction algorithm minimizes the error from the raw data using the interpolated three-dimensional (3D) plane and convolution method.

![Figure 1. Concept of the LiDAR signal weight reduction using data reduction based on time variation and reconstruction.](image)

2. Related Work

Recently, LiDAR sensors have been widely studied and various applications have been actively introduced [3]. In particular, research on LiDAR sensors for autonomous vehicles is focused on industrial applications [4] due to the depth-based 3D reconstruction capability. LiDAR sensors are being used for 3D precise mapping and object detection, recognition, and ranging for autonomous driving [5,6]. For that, it is necessary to mount multiple LiDAR sensors, and research involves a system for processing multiple LiDAR
sensors simultaneously or for integrated object detection and recognition through data matching between different types of sensors, such as camera and radar [7,8]. LiDAR manufacturers are also developing products for this purpose [9,10]. Currently, the NVIDIA Jetson Xavier platform with computing power and GPU parallel processing is widely used for object detection and recognition using LiDAR sensors [11,12]. Our previous study was for real-time processing of LiDAR data on an embedded system with low power and multiple LiDAR sensor data input [13–15]. Using this embedded system, we continued the study on the calibration to reduce the error between LiDAR data during localization using multiple LiDAR sensors and the study on the LiDAR data reduction algorithm through a frame-to-frame comparison [16,17].

Our previous work introduced an on-demand remote code execution-based acceleration platform to increase the computational speed of embedded devices. The acceleration platform reduces the program execution time by separating the program to be executed on the embedded device and the program to be executed on the server according to the amount of transmission data [18]. The metamorphic platform that accelerates programs by reconfiguring the accelerator of embedded devices in real-time was introduced [19,20]. In this paper, based on previous research, we compressed and transmitted the data received from multiple LiDAR sensors, received it from an external embedded system, and restored it close to the original data using only reduced data. We fully implemented LiDAR-based embedded systems, including a LiDAR sensor front-end, a data signal processing unit, and a 3D object reconstruction engine to evaluate our proposed scheme, hardware–software platform toward the energy-efficient self-driving algorithm-executable platform.

3. Proposed Method

3.1. LiDAR Sensor

In this paper, we used the VL-AS16 LiDAR sensor with a 16-channel laser detector and a laser emitter. The VL-AS16 LiDAR sensor, which generated 1160 horizontal data points, had a horizontal FOV of 145° and a horizontal angular resolution of 0.125°. Owing to the 16 channels of the laser detector, it had a vertical FOV of 9.6° and a vertical angular resolution of 0.6°, as shown in Figure 2.

![Figure 2](image-url)  
**Figure 2.** The shape of the point cloud detected by the VL-AS16 LiDAR sensor and the distance data accumulated over time.

The one-point raw data of the LiDAR sensor is expressed as the 2-byte distance and 1-byte intensity value, which represents the return strength of the laser according to the transmission protocol. Therefore, the LiDAR sensor with 16 channels, which include 1160 horizontal data points, produces 55,680 bytes of data in one scan. If the LiDAR sensor
is mounted on a vehicle, it has to scan in all directions, and needs to mount multiple sensors. Considering that the FOV of VL-AS17 is $145^\circ$, at least four sensors should be mounted on the front, rear, left, and right sides of the vehicle. To process the raw data of VL-AS16 LiDAR with a 30-Hz scan cycle in real-time, the processor had to process 55.6 KB of data in 33 ms. Real-time processing of the vehicle, which used four LiDAR sensors to detect all quarters, had to process 223 KB of data in 33 ms.

The scanned distance data are expressed as frame data in which each point has a distance value. When frame data detected in multiple scans are arranged on the time axis, the frame data of the adjacent time tend to have a large number of point clouds with similar distance values. Compared with the frame data of the adjacent time, the difference between the distance data of the background is small, and the difference between the distance data of the object is large. In this paper, we propose a platform that reduces the amount of transmission data between the LiDAR sensors and an external embedded processor using semantic depth data, highlighted by comparing the frame data detected by the LiDAR sensors on the time axis.

3.2. Integrated Processing Unit

The integrated processing unit (IPU) reduces the amount of data transmitted to the external system by compressing the raw data sensed by the four LiDAR sensors [21]. The IPU system, as shown in Figure 3, consists of one server process and multiple client processes. The four client processes of the IPU run in parallel to acquire and process the raw data from the four LiDAR sensors [15]. The raw data produced by LiDAR every 33 ms are delivered to the client process of the IPU using the user datagram protocol (UDP). Each client removes unnecessary parts of the raw data, processes the data according to a reduction algorithm, and aligns the processed data according to the protocol. The server process receives processed data and adds the header, number of sensors, data size, checksum, and tails to fit the protocol, as shown in Table 1.

![Figure 3. The software execution flow based on the server-client process in the integrated processing unit.](image-url)
Table 1. The LiDAR data transfer protocol, including sensor information.

| Position | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|----------|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|
| Name     | Header | Sensor number | Data size | Sensor ID | Data type | X-coordinate | Y-coordinate | Z-coordinate |
| Position | 15 | 16 | 17 | 18 | 19 | 20 | 21 | N | N+1 | N+2 | N+3 | N+4 |
| Name     | Roll | Pitch | Yaw | 0 | Packet data | Checksum | Tail |

For a data packet inside the protocol, the method of parsing the data varies according to the transmission mode. In LiDAR transmission, three data packet modes are used: raw data, length, and coordinate. In the proposed IPU structure, the length mode was used for data transmission. In the IPU using the length mode, the client arranges the raw LiDAR data in the data packet format presented in Table 2. Moreover, 1-point ‘data’ consist of a 2-byte distance value and a 1-byte intensity value of the laser. Based on the point data obtained from the 16 channels, one data packet was composed of the point data acquired from each of the 16 channels and the order information of the current data packet. The horizontal angle represents the sequence of data packets from 0 to 1159, and the distance value indicates the value calculated by the LiDAR sensor converted into cm units. Behind the horizontal angle value, the distance value calculated by the sensor and the laser intensity value are listed from channels 0 to 15. Thus, a one-angle data packet consists of 50 bytes. According to the protocol, 59.2 KB of LiDAR raw data are processed into 58 KB of packet data.

Table 2. The packet data bytes of the LiDAR protocol configured in length mode.

| Byte Number | 22 | 23 | 24 | 25 | 26 | ... | 69 | 70 | 71 |
|-------------|----|----|----|----|----|-----|----|----|----|
| Name        | Horizontal Angle | Distance Value | Intensity | Distance Value | Intensity |
| Value       | 0 to 1159 | 0 to 25,000 | 0 to 255 | 0 to 25,000 | 0 to 255 |

Table 3 lists the partial-length mode data packets used in the IPU using the reduction algorithm. The client process of the IPU uses a reduction algorithm to reduce the size of the length-mode data packets. The reduced protocol transmits 1160 packets based on the horizontal angles and determines the horizontal angles through sequentially-transmitted packets. A data packet in the partial-length mode has a 2-byte channel on/off flag. The 16 bits of the channel’s on/off flag indicate whether the channel data are displayed in the partial length mode. In the VL-AS16 LiDAR protocol, the most significant bit of the channel’s on/off flag indicates the 15th channel’s on/off and the least significant bit indicates the 0 channel’s on/off. After the channel’s on/off flag, the distance and intensity values of the enabled channel are added.

Table 3. The packet data bytes of the LiDAR protocol configured in the partial length mode.

| Byte Number | 22 | 23 | 24 | 25 | 26 | ... | 22 + Nch \times 3 | 23 + Nch \times 3 | 24 + Nch \times 3 |
|-------------|----|----|----|----|----|-----|----------------|----------------|----------------|
| Name        | Channel On/Off | Distance Value | Intensity | Distance Value | Intensity |
| Value       | 0 to 0xFFFF | 0 to 25,000 | 0 to 255 | 0 to 25,000 | 0 to 255 |

3.3. Reduction Algorithm

We targeted a vehicle system that uses four LiDAR sensors to detect all quarters. One LiDAR sensor produces 55.6 KB of data after 30 cycles. Therefore, an existing system using four LiDAR sensors transmit and process 232 KB of data in 33 ms. It is difficult to
process a large amount of data in a vehicle system equipped with an embedded processor in real-time. To reduce the amount of data transmitted by the LiDAR, we propose a data reduction algorithm based on the time variation of the detected distance. Figure 4 shows a flowchart of the reduction algorithm executed in the LiDAR sensor and the IPU system.

Figure 4. Reduction algorithm execution flowchart of the LiDAR and integrated processing unit system: (a) LiDAR sensor. (b) Client process of the IPU. (c) Server process of the IPU.

Figure 4a shows the behavior of the LiDAR sensor. The entire data detected by the LiDAR sensor in one scan, as denoted by frame \( F \), takes the form of a two-dimensional (2D) array with the number of rows by the number of LiDAR channels and the number of columns by the LiDAR horizontal angular resolution. The LiDAR sensor detects 16 channels of distance data, denoted by packet \( P_i \) in one laser emission. One-frame of data \( F \) is produced by collecting 1160 LiDAR detection data. The data produced are delivered to the client processes of the IPU through a UDP connection.

The IPU consists of client processes, which are connected to each LiDAR sensor, process data, and a server process, which collects data from clients and transmits them to the external system. Figure 4b shows the operation of the IPU client process. One LiDAR sensor scans the same location repeatedly. When the data scanned by the sensor are transmitted to the client process, the previous frame is saved as the previous frame, and the transmitted data are saved as the current frame. The client process extracts the points of semantic depth data, as shown in Figure 5, calculating the difference between each point of the current frame data and the previous frame data. If the change of distance data between frames of each point exceeds the threshold, the point is regarded as a semantic depth point and the data of the current frame are updated in the transmission frame.

Figure 5. Extracting semantic depth data points by comparing frames over time.

The calculated error data \( e_{i,j} \) are compared with the distance threshold \( d_{th} \). If the error data are greater than the threshold, the data are preserved in the error packet \( E_i \) and the check flag \( c_{i,j} \) of the same coordinates is set, because \( e_{i,j} \) represents the edge of the object. Error data smaller than the threshold are regarded as data with low importance and are deleted from the error packet \( E_i \), and \( c_{i,j} \) is set to 0. In this study, we set the threshold...
value $d_{th}$ to 15 cm, which is the distance that the light travels in 1 ns from the LiDAR sensor. The client process updates the previous frame to the current frame and passes the reduced error data packet $E_i$ and the check flag array $C_i$ (which indicates whether the frame point data are removed) to the server process.

The server process receives the check flag array $C_i$ and error data packet $E_i$ from the client processes and transmits the data to the external system according to the protocol, as shown in Figure 4c. The information of the LiDAR sensor is inserted into the header of the protocol presented in Table 1, and the reduced data are transmitted as a check flag $C_i$ and error data packet $E_i$ as shown in Table 3. The server process sequentially transmits reduced data packets out of the total data, so that the external system can compose one frame using the transmitted packets.

The data packet size of the IPU system without using the reduction algorithm is 58,026 bytes. In one frame scanned by the LiDAR sensor, all points—except the region of interest (ROI), where the object was detected—were zero, indicating a background value. As the LiDAR sensor scanned the object in free space, the ROI was less than half. Therefore, the size of the partial data packet transmitted by the IPU system using the reduction algorithm was reduced by more than half.

### 3.4. Reconstruction Algorithm

The IPU system transmits data compressed by the time-change-based reduction algorithm to the external system. Because the reduced data have only the minimum data for recognizing the shape of an object, it is necessary to restore the original data to execute additional algorithms, such as object recognition. Restoring the reduced data to the original data can be divided into (1) a method that uses reference data and (2) a method that does not use reference data. In restoring the original data using reference data, the IPU transmits the original data that are not reduced in every predetermined number of times and restores the reduced data based on the reference data. However, the reference data-based restoration method increases the data transmission time and memory demand by storing the reference data. In this study, a restoration method without reference data was used, assuming that only reduced data were transmitted.

Figure 6 shows the process of restoring the reduced data in an external system. The data reconstruction algorithm consists of a reduced frame construction step with a packet reception, a distance grouping step using semantic depth data, and a frame reconstruction step using spatial similarity. The external system received reduction-mode packets from the IPU, and was composed of a reduced frame. The reduced frame was a 2D array in which the remaining points (except for the valid data) were filled with zeros by the check flag. To execute the reconstruction algorithm, one frame array was divided into 16-channel array data $D$. The channel array data $D$ were compensated for the data removed inside the object in the distance grouping step, as shown in Figure 7.

Figure 6b shows the distance grouping step of the reconstruction algorithm, which receives and executes one channel of frame $D$ as an input. The distance grouping step was divided into a grouping with similar distance information in the channel, finding the start and endpoints of an object using a similar distance group, and reconstruction smoothing. Algorithm 1 shows the distance grouping step. In the decision data classification step, similar data were extracted by comparison with a basic length set $B$ for all valid data in $D$. When the difference between valid data $d_i$ and basic length data $b_i$ exceeded a threshold $d_{th}$, data $d_i$ were added to the basic length set $B$. After the decision data classification, values excluding similar data from the one-channel data were stored in the basic length set $B$. In the index grouping step, indices of similar data were grouped using the basic length set $B$. Set $I_g$ had indices for data with a distance value similar to each element in the basic length set $B$. Set $G$ had $I_g$ sets for each basic length set element (as elements).

Algorithm 2 finds the start and end points of the edge of the object using the grouped indices $G$. Because the distance points of one object have similar values, the points constituting the object can be discovered by the elements of $I_g$. Algorithm 2 executes on all $I_g$ sets.
in G. The first element of $I_g$ is added to the reconstruction start set $I_{start}$. For all elements of $I_g$, if the difference between the current element and the previous element exceeds the recognition index threshold $p_{th}$, the previous element is added to the end set $I_{end}$ and the current element is added to the start set $I_{start}$. The distance grouping index threshold $p_{th}$ is the minimum distance between the edges of the recognized object. Using this algorithm, the set of start points $I_{start}$ and the set of end points $I_{end}$ in each channel are calculated.

![Flowchart](image)

**Figure 6.** Reconstruction algorithm execution flowchart in the external system. (a) Data reception and frame construction. (b) Distance grouping for each channel. (c) Frame reconstruction using convolution.

![Matrix](image)

**Figure 7.** Concept of the point cloud change in the distance grouping step for each channel.

Algorithm 3 shows the process of refining the distance data of the inner point of each channel array $D$ using the start and end points of the edges obtained in Algorithm 2. If the $i$-th element of the reconstruction start point set $I_{start}$ is denoted by $s_i$ and the $i$-th element of the reconstruction end point set $I_{end}$ is denoted by $e_i$, the algorithm is executed for all $s_i$ and $e_i$ pairs. The value of point $d_j$ between the $s_i$ and $e_i$ points varies linearly. For example, when $j$ is $s_i$, $d_j$ is equal to the value of $d_{s_i}$, and when $j$ is $e_i$, $d_j$ is equal to the value of $d_{e_i}$. 
Algorithm 1 Index parsing algorithm with similar distance.

1. $D = \{d_i | 0 \leq i \leq 1159\}$: Channel array data.
2. $B = \{b_i \in B\}$: Set of basic length.
3. $I_g$: Set of indices with similar distances.
4. $G$: Set of $I_g$.
5. $d_{th}$: Distance similarity threshold.
6. % Classification of decision data
7. $l = d_0$
8. foreach $d_i$ in $D$ do
9.  if $|d_i - l| > l \times d_{th}$ then
10.     foreach $b_j$ in $B$ do
11.         $c = 1$
12.         if $|b_j - d_i| < b_j \times d_{th}$ then
13.             $c = c \& 0$
14.         if $c$ is 1 then
15.             $l = d_i$
16.             $B \ni d_i$

17. % Index grouping
18. foreach $b_i$ in $B$ do
19.     foreach $d_j$ in $D$ do
20.         if $|b_i - d_j| < b_i \times d_{th}$ then
21.             $I_g \ni j$
22.     $G \ni I_g$

Algorithm 2 Calculate the reconstruction start and end points.

1. $I_g$: Set of indices with similar distances.
2. $G$: Set of $I_g$.
3. $I_{start}$: Set of reconstruction start indices.
4. $I_{end}$: Set of reconstruction end indices.
5. $p_{th}$: Distance grouping index threshold.
6. foreach $I_g$ in $G$ do
7.     $p = 0$
8.     foreach element $i$ in $I_g$ do
9.         if $i$ is first element of $I_g$ then
10.             $I_{start} \ni i$
11.         else if $i$ is last element of $I_g$ then
12.             $I_{end} \ni i$
13.         else
14.             if $i \geq p + p_{th}$ then
15.                 $I_{end} \ni p$
16.                 $I_{start} \ni i$
17.                 $p = i$
Algorithm 3 Reconstruction smoothing.

1. $D$: Channel array data.
2. $I_{\text{start}}$: Set of reconstruction start indices.
3. $I_{\text{end}}$: Set of reconstruction end indices.
4. $s_i$: $i$-th element in $I_{\text{start}}$.
5. $e_i$: $i$-th element in $I_{\text{end}}$.
6. foreach $e_i$ in $I_{\text{end}}$ do
   7. for $j$ from $s_i$ to $e_i$ do
      8. $d_j = \frac{(j-s_i) \times d_{e_i} + (e_i-j) \times d_s}{e_i-s_i}$

Multiple edge information points can be included in one channel of the point cloud. The edge information of the object included in the reduced data can be added or deleted by the scanning noise of the sensor and transmitting noise. In the distance grouping stage, because the edge of the object is reconstructed using the emphasized reduced data, an error occurs in the noise. In the process of grouping the edges with similar distances, the index threshold $p_{lh}$ between the edges prevents an error in which one edge is detected as multiple edges owing to noise. An object with many points inside the edge is reconstructed with a small error using the $p_{lh}$ of the distance grouping stage. However, for objects with a few points inside the edge, the reconstruction error increases.

Figure 6c shows the process of frame reconstruction using convolution after reconstructing an object using an edge internal point reinforcement algorithm. In the point cloud of the scanned LiDAR data, one object has similar distance data between adjacent points. In the frame transformed by the reduction algorithm, edge information is displayed at a similar point in an adjacent channel. In the frame reconstruction algorithm, the error in edge information due to noise is reduced by the convolution of adjacent channels. The overall process is illustrated in Figure 8.

Figure 8. Concept of the point cloud change in the frame reconstruction step using convolution.

4. Experiment and Measurement Results

For the experimental environment, one NXP LS1028A board and four VL-AS16 LiDAR sensors were configured, as shown in Figure 9 [22]. The Ethernet port of the LS1028A board was connected to the four LiDAR sensors and one external system to configure the proposed IPU system. The client and server of the IPU system were distributed across the two ARM Cortex-A72 cores of the LS1028A board and executed in parallel. To express the data produced by the four LiDAR sensors in a 2D graph, the sensors were set up vertically as shown in Figure 10a. The experimental data were used to measure the movement of a person in the FOV of the sensors set in free space, as shown in Figure 10b.
The raw data of LiDAR sensors are usually visualized as a 3D point; however, in this study, the 3D point was expressed as a 2D point projected onto the x–z-plane. In the 2D graph, the x-axis represents the horizontal FOV of the LiDAR and the z-axis represents the vertical FOV. The distance to the object was expressed as the value of each point. The laser emitted by the LiDAR sensor spread out in a fan shape from the sensor. Therefore, a close object was projected with a shorter height and a distant object was projected with a longer height [23]. Figure 11 shows the results of presenting the data detected by the sensor in a 2D plane, as shown in Figure 10b. To distinguish the shape of the object in the 2D graph, points having a distance value of less than four are colored black, and points with distance values greater than four are colored gray.

![Figure 9](image-url)  
**Figure 9.** Experiment environment of VL-AS16 LiDAR sensors and th LS1028A-integrated processing unit board.

![Figure 10](image-url)  
**Figure 10.** Physical experiment environments. (a) Positioning of the four LiDAR sensors. (b) The picture of a moving object (person) detected by the LiDAR sensors.
4.1. Reduction

In the experimental environment, the LiDAR sensor measured a person at a distance of approximately four m from the sensor. Figure 12a shows the data detected by one LiDAR in a 2D graph, and Figure 12b shows the data detected by the four LiDAR sensors in one graph. The raw data from the LiDAR sensor were composed of data representing the object and those representing the background. Figure 12c and Figure 12d show the data from one LiDAR sensor and four LiDAR sensors, respectively, after executing the reduction algorithm. As a result of executing the reduction algorithm, the number of points representing the background and the number of points inside the object were reduced. Compared to the original point cloud, the reduced point cloud output the edge data of the object [24]. The reduced data of one LiDAR sensor, as shown in Figure 12c had sufficient data to distinguish the human shape; however, the reconstruction accuracy was reduced because it had less information compared to the original data.

![Figure 11. Point cloud of a moving object (person) in an experimental environment.](image)

![Figure 12. Point cloud data. (a) Full data from one LiDAR sensor. (b) Full data from four LiDAR sensors. (c) Reduced data from one LiDAR sensor. (d) Reduced data from four LiDAR sensors.](image)

One point of the LiDAR sensor was composed of a distance value of two bytes and a laser intensity value of one byte. Therefore, 1 data packet, composed of 16 points and a horizontal angle value of 2 bytes, contained 50 bytes of data. Because one 2D frame consisted of 1160 data packets, the LiDAR sensor transmitted 58,000 bytes of data in one scan. Figure 13 shows the results of reducing the data obtained from 300 scans using one LiDAR sensor. The four LiDAR datasets reduced a similar amount of data for the same object; however, the amount of transmitted data differed owing to the difference in the position of the sensor. As a result of the reduction algorithm, the data were reduced to a maximum of 10,143 bytes, a minimum of 5310 bytes, and an average of 7295 bytes.
Compared with the raw data, the reduction algorithm reduced the amount of data by up to 89.6%, at least 85.8%, and on average by 87.4%.

Figure 13. Amount of transmitted data reduced by the proposed algorithm. (a) Overall number of transmitted data. (b) Detailed view of the red-colored box region.

We used the Valgrind program to measure the memory usage of the IPU [25]. Figure 14 shows the memory usage of the client and server processes running on the IPU. Figure 14a shows the memory usage of the client process that transmitted the raw LiDAR data, and Figure 14c shows the memory usage of the client process that transmitted the LiDAR data after reduction. The client process executed the data produced by one LiDAR and transmitted them to the server process according to the protocol. The client process without data reduction used an average of 1.48 MB of memory and took 739.15 s to transmit the data. The client process that executed the data reduction algorithm used an average of 1.41 MB of memory and took 441.0 s to transmit the data. The reduction algorithm of the client process reduced the size of the LiDAR data, thereby reducing the memory usage and transmission time. Figure 14b and Figure 14d show the memory usage of the server process without and with the reduction algorithm, respectively. For the data transmission produced by the four LiDAR sensors, the server process without and with the reduction algorithms used an average of 319 kB and 219 kB of memory, respectively. The server process with the reduction algorithm had reduced memory usage because the amount of data transmitted was reduced compared with the previous one.

Figure 14. Memory usage of the integrated processing unit, (a) Client without reduction. (b) Server without reduction. (c) Client with reduction. (d) Server with reduction.

4.2. Reconstruction

The data produced by the LiDAR sensor VL-AS16 was reduced by the algorithm loaded on the LS1028A IPU board. The reduced data were transmitted to the LX2160A board, which was an external system, through an Ethernet connection. We used a LX2160A board as an external system for an experiment similar to the vehicle’s embedded processor. Figure 15 shows the point cloud when the reduced data were restored using the reconstruc-
tion algorithm. The data reduced in the IPU were reconstructed using distance grouping and frame reconstruction before being used in the external system.

![Figure 15. Point cloud data in person ROI. (a) Original full data (b) Reduced data (c) Results of distance grouping. (d) Results of frame reconstruction with a convolution filter.](image)

Equation (1) describes the reconstruction error in the edge-based distance grouping and convolution-based frame reconstruction steps. The ROIs of raw LiDAR data, reconstruction data after distance grouping, and frame reconstruction data were filtered using the characteristic at which the background data appeared $\leq 0$. $N$ was the number of points inside the ROI of the point cloud and was equal to 3200 points composed of 200 packets with 16 channels. $d_i$ was the distance value of the $i$-th point, and $d_i^{\text{recon}}$ was the distance value of the reconstructed $i$-th point. The total error $E$ was obtained by dividing the number of points $N$ by the sum of the errors, based on the mean absolute error. To correct the error of points with different distance values, the value obtained by dividing the absolute value of the difference between $d_i$ and $d_i^{\text{recon}}$ by $d_i$ was set as the error of each point.

$$E = \frac{1}{N} \sum_{i=0}^{N} \frac{|d_i - d_i^{\text{recon}}|}{d_i}$$  \hspace{1cm} (1)

Figure 15 shows the point cloud of person–object ROI. Figure 15a shows the point cloud of the original full data from the LiDAR sensor. Figure 15b shows the point cloud of the reduced data in the IPU system using the reduction algorithm. Figure 15c shows the point cloud when the distance grouping algorithm was executed. In the distance grouping step, the distance data inside the edge of the object were filled. However, depending on the number of edges expressed by the reduction algorithm or the amount of data loss owing to the shape of the object, the internal data of the object not filled with similar data or data between objects were filled. Comparison between raw data and length-grouping-based reconstruction data from the four LiDAR sensors using Equation (1) resulted in errors of 8.60%, 31.06%, 31.70%, and 35.66%. Data reconstruction based on length grouping had low similarity between the data inside the object and those at the edge and recognized a nearby object as a single object.

Figure 15d shows the point cloud of a reconstruction algorithm using the convolution filter. To solve the problem of the length-grouping-based object reconstruction algorithm, convolution strengthens the continuity of objects. A comparison between the raw data and convolution-based reconstruction algorithm data from the four LiDAR sensors using Equation (1) result in errors of 1.70%, 4.37%, 6.55%, and 4.93%. When the reconstruction data with length-groupings were supplemented based on convolution, the similarity between the internal and edge data of the object increased, and adjacent objects were divided.

The external system reconstructed the reduced data and displayed them on screen. To compare the reduction algorithm with an existing platform, we changed the IPU behavior of the proposed platform. The IPU of the existing platform, which transmitted raw data, bypassed LiDAR sensor data to the external system. The proposed platform, including the
reduction and reconstruction algorithm, reduced the data of the LiDAR sensors in the IPU and reconstructed them in the external system.

Figure 16 shows the memory usage when the data transmitted from the IPU were restored from the external system and displayed on the screen. In the experiment, 1244 frames of data generated by each LiDAR sensor were transmitted to the IPU. The IPU collected data from the four LiDAR sensors and sent them to the external system. Figure 16a shows the memory usage of the external system when the IPU transmitted raw data from the LiDAR sensors. When the IPU transmitted uncompressed raw data, the external system received the data and output them using OpenCV. To process the received LiDAR data, the external system took 67.221 s and used an average of 21.61 MB of memory. Figure 16b shows the memory usage of the external data when the IPU transmitted the reduced data in the proposed platform. The data reduced by the IPU were restored using the reconstruction algorithm in the external system and then displayed on the screen. To receive and process the reduced data, the external system took 73.307 s and used an average of 28.74 MB of memory.

![Figure 16. Memory usage of the external system. (a) Full data transmission. (b) Reduced data transmission and reconstruction.](image)

Figure 16. Memory usage of the external system. (a) Full data transmission. (b) Reduced data transmission and reconstruction.

Figure 17 shows the time to transmit one frame of data from the IPU to the external system. Figure 17a shows the time taken when sending raw and reduced data from the IPU, and Figure 17b shows the time taken for the external system to receive raw and reduced data. Without the reduction algorithm, the IPU took an average of 0.20 ms to transmit one frame, and the external system took an average of 1.79 ms to receive one frame. In the proposed platform, the IPU took an average of 0.035 ms to transmit one frame, and the external system took an average of 0.28 ms to receive one frame.

![Figure 17. Frame communication time between IPU and the external system. (a) IPU execution time per frame when transmitting raw and partial data. (b) External system execution time per frame when receiving raw and receiving partial data.](image)

Figure 17. Frame communication time between IPU and the external system. (a) IPU execution time per frame when transmitting raw and partial data. (b) External system execution time per frame when receiving raw and receiving partial data.

5. Conclusions

This study proposes a lightweight LiDAR signal processing platform to reduce the data transmission amount for LiDAR sensors in the embedded environment of vehicles with limited computational resources. The LiDAR sensors used in vehicles continuously observe and transmit data to the main processor. In continuous data transmissions, most of the currently transmitted data consist of point data similar to previous data. In this study, the sizes of the LiDAR sensor data were reduced by using a reduction algorithm that extracted semantic depth data and compared changes with the LiDAR sensor data over time. The reduced LiDAR sensor data were transmitted in a structure using the partial-length mode of the data transmission protocol. The raw data (with 58,026 bytes produced by one LiDAR sensor) were reduced by approximately 87.4% to an average of 7295 bytes using the data reduction algorithm of the IPU. Compared to an existing
system that transmits raw LiDAR data, the IPU using the reduction algorithm increased the memory usage of the server process by 100 kB and the memory usage of the client process by 70 kB; however, the execution time decreased by 40.3% owing to the reduction in the data transfer time.

The reduction data, which emphasize the edge data of the object, are restored before use in the main processor of the vehicle. The data reconstruction algorithm converts the data reduced by the distance grouping, and frame reconstruction steps into data that can be used for object recognition in the main processor. The distance grouping step restores data inside the object edge by grouping similar data from the reduced data. Distance grouping based on length grouping is effective at restoring reduced data inside the object; however, when the size of the object is small or the distance between the objects is narrow, the error with the original data increases. To reduce the error in distance grouping with reconstruction smoothing, the data reconstruction algorithm uses convolution-based frame reconstruction. The frame reconstruction step increases the similarity of the data inside the object through a convolution operation between adjacent channels in one frame. Comparing the ROI of the original data with the data reconstructed by the algorithm, the distance grouping and frame reconstruction steps resulted in average errors of 26.76% and 4.39%, respectively.

The proposed lightweight LiDAR signal processing platform uses reduction and reconstruction algorithms to reduce the amount of data transmission between the LiDAR sensors and the vehicle’s main processor. Data reduction of IPU increases memory usage compared to raw data transmission systems and requires additional time to execute the reduction algorithm. However, the total data transfer time is substantially reduced, thereby reducing the overall execution time. The main processor of the vehicle receiving the reduced data performs the object recognition and the data recovery algorithms. The IPU of the proposed platform took 0.28 ms to receive one frame (58,000 bytes) produced by the LiDAR sensor, which was reduced from 1.79 ms in the existing system. With the reconstruction algorithm, the main processor’s overall execution time was 67.474 s, up from 63.217 s, with raw data transmission. Based on this study, we will study to improve the LiDAR data reduction algorithm and reconstruction algorithm through LiDAR data in various environments, such as outdoor free space.

Author Contributions: T.C. presented the entire model platform, and designed, implemented, evaluated the experimental design; D.L. organized and analyzed the experimental results and contributed to the writing of the paper; D.P. presented the basic concept of the paper and was the principal investigator and corresponding author. All authors have read and agreed to the published version of the manuscript.

Funding: This work supported by the Technology Innovation Program (P0013847, development of the automatic steering-based accident avoidance system for electric-driven port yard tractors operating at low speed (less than 30 km/h), 50%) funded by the Ministry of Trade, Industry, and Energy (MOTIE, Korea), and partially supported by the BK21 FOUR project funded by the Ministry of Education, Korea (4199990113966, 10%), and the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2018R1A6A1A03025109, 10%). This work was partly supported by an Institute of Information and communications Technology Planning and Evaluation (IITP) grant funded by the Korean government (MSIT) (no. 2021-0-00944, metamorphic approach of unstructured validation/verification for analyzing binary code, 10%), (no. 2022-0-00816, OpenAPI-based hw/sw platform for edge devices and the cloud server, integrated with the on-demand code streaming engine powered by AI, 10%), and (no. 2022-0-01170, PIM Semiconductor Design Research Center, 10%).

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.
Abbreviations
The following abbreviations are used in this manuscript:

- LiDAR: light detection and ranging
- IPU: integrated processing unit
- ROI: region of interest
- FOV: field of view
- Radar: radio detection and ranging
- laser: light amplification by stimulated emission of radiation
- UDP: user datagram protocol

References

1. Haq, M.A.; Ruan, S.J.; Shao, M.E.; ul Haq, Q.M.; Liang, P.J.; Gao, D.Q. One Stage Monocular 3D Object Detection Utilizing Discrete Depth and Orientation Representation. *IEEE Trans. Intell. Transp. Syst.* 2022, 1–11. [CrossRef]
2. Ul Haq, Q.M.; Haq, M.A.; Ruan, S.J.; Liang, P.J.; Gao, D.Q. 3D object detection based on proposal generation network utilizing monocular images. *IEEE Consum. Electron. Mag.* 2021, 13, 8–14. [CrossRef]
3. Maksymova, I.; Steger, C.; Druml, N. Review of LiDAR sensor data acquisition and compression for automotive applications. *Multidiscip. Digit. Publ. Inst. Proc.* 2018, 2, 852.
4. Sualeh, M.; Kim, G.W. Dynamic Multi-LiDAR Based Multiple Object Detection and Tracking. *Sensors* 2019, 19, 1474. [CrossRef] [PubMed]
5. Giacalone, J.P.; Bourgeois, L.; Ancora, A. Challenges in aggregation of heterogeneous sensors for Autonomous Driving Systems. In Proceedings of the 2019 IEEE Sensors Applications Symposium (SAS), Sophia Antipolis, France, 11–13 March 2019; pp. 1–5.
6. Bouain, M.; Ali, K.M.; Berdjag, D.; Fakhfakh, N.; Atitallah, R.B. An Embedded Multi-Sensor Data Fusion Design for Vehicle Perception Tasks. *J. Commun.* 2018, 17, 525–534. [CrossRef]
7. D’Angio, P.C. Adaptive and Passive Non-Visual Driver Assistance Technologies for the Blind Driver Challenge. Ph.D. Thesis, Virginia Tech, Blacksburg, VA, USA, 2012.
8. NXP. LS1028A Data Sheet. Available online: https://www.nxp.com/products/microcontrollers-arm-processors/layerscape-processors/layerscape-processors/layerscape-1028a-applications-processor:LS1028A (accessed on 1 May 2022).
23. Gupta, S.; Lohani, B. Augmented reality system using lidar point cloud data for displaying dimensional information of objects on mobile phones. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* 2014, 2, 153. [CrossRef]

24. Haq, Q.M.U.; Lin, C.; Ruan, S.J.; Gregor, D. An edge-aware based adaptive multi-feature set extraction for stereo matching of binocular images. *J. Ambient. Intell. Humaniz. Comput.* 2022, 13, 1–15. [CrossRef]

25. Nethercote, N.; Seward, J. Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation. *ACM Sigplan Not.* 2007, 42, 89–100. [CrossRef]