Received November 6, 2021, accepted December 28, 2021, date of publication January 4, 2022, date of current version January 18, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3140332

Spatial-Importance-Based Computation Scheme for Real-Time Object Detection From 3D Sensor Data

RYO OTSU[1], RYOICHI SHINKUMA[2], (Senior Member, IEEE), TAKEHIRO SATO[1], (Member, IEEE), EIJI OKI[1], (Fellow, IEEE), DAIKI HASEGAWA[3], AND TOSHIKAZU FURUYA[4]
[1]Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan
[2]Faculty of Engineering, Shibaura Institute of Technology, Tokyo 135-8548, Japan
[3]ExaWizards Inc., Tokyo 105-0021, Japan
[4]Quantum Analytics, Inc., Hiroshima, 737-0821, Japan

Corresponding author: Ryoichi Shinkuma (shinkuma@shibaura-it.ac.jp)

This work was supported in part by the JST PRESTO under Grant JPMJPR1854; and in part by the JSPS KAKENHI, Japan, under Grant 21H03427 and Grant 21H03426.

ABSTRACT Three-dimensional (3D) sensor networks using multiple light-detection-and-ranging (LIDAR) sensors are good for smart monitoring of spots, such as intersections, with high potential risk of road-traffic accidents. The image sensors must share the strictly limited computation capacity of an edge computer. To have the computation speeds required from real-time applications, the system must have a short computation delay while maintaining the quality of the output, e.g., the accuracy of the object detection. This paper proposes a spatial-importance-based computation scheme that can be implemented on an edge computer of image-sensor networks composed of 3D sensors. The scheme considers regions where objects exist as more likely to be ones of higher spatial importance. It processes point-cloud data from each region according to the spatial importance of that region. By prioritizing regions with high spatial importance, it shortens the computation delay involved in the object detection. A point-cloud dataset obtained by a moving car equipped with a LIDAR unit was used to numerically evaluate the proposed scheme. The results indicate that the scheme shortens the delay in object detection.

INDEX TERMS Smart monitoring, object detection, LIDAR sensor, point cloud, edge computing.

I. INTRODUCTION

The smart cities of tomorrow are expected to use smart monitoring, particularly at intersections, to prevent traffic accidents [1]. As Datondji et al. suggested [2], safety at intersections is a critical global issue, as accidents at intersections are a major cause of road fatalities. Smart monitoring using three-dimensional (3D) sensors, in particular, using light detection and ranging (LIDAR) sensors, is a promising means to prevent such accidents [3], [4]. In particular, a smart-monitoring system using multiple 3D sensors would be effective for detecting objects such as cars, cyclists, and pedestrians; in contrast, if a single 3D sensor only is used, target objects can be easily occluded by foreground objects.

A 3D-object detector takes a point cloud of a scene as input and produces an oriented 3D bounding box around each detected object [5]. In smart monitoring, it is necessary for the computation delay in object detection to be short as well as for the detection to be accurate. However, the previous studies on object detection using point clouds of scenes focused mainly on the accuracy of the detection without considering the computation time [5].

Applying edge computing to smart monitoring is straightforward and has an advantage in terms of reducing communication latency between a network edge and a cloud computer [6]. However, it is not easy to perform object detection in real time with edge computers because their computation power is limited. Typically, LIDAR sensors are operated at frequency of 5 to 20 Hz, meaning they generate 5 to 20 frames per second. As we will show later, even using a cloud server, the computation time of object detection on 3D sensor data can be up to 200 to 300 milliseconds per frame. As such, an edge computer might receive the next frame of 3D data before it has finished processing the preceding

The associate editor coordinating the review of this manuscript and approving it for publication was Junhua Li.

frame, meaning that it can't process the 3D data in real time.

This paper proposes a spatial-importance-based computation scheme that works on the edge computers of 3D sensor networks composed of multiple LIDAR units. The scheme splits up a point cloud into regions according to the spatial importances of those regions; for example, regions with objects such as cars, trucks, bicycles, and pedestrians are more likely to be important ones for safety monitoring. The points in these regions are then processed sequentially or in parallel. Suppose that a point cloud has been split into a number of high-importance regions and low-importance regions. In the case of sequential processing, the data of the high-importance regions are processed first; this shortens the computation time until a certain number of objects are detected. In the parallel case, the computation on the high-importance regions will likely finish earlier than that of the low-importance regions, which tends to contain a lot of data on roads, buildings, etc., in the background. In either case, the delay in obtaining the object detection result is shortened. In this study, we assume that a machine-learning (ML) model that is pre-trained on a cloud computer is used to perform object detection on the edge computer. We used a real point-cloud dataset collected by a moving car equipped with a LIDAR unit to numerically evaluate the proposed scheme. The results indicate that the proposed scheme shortens the computation time while meeting the accuracy required for object detection.

As a means of overcoming the bandwidth limitation issue, the authors previously presented schemes that differentiate the compression rates of point cloud data in different spatial regions in accordance with an importance score when the data are transmitted from sensors to an edge computer [4], [7]. In this paper, we address the processing time involved in object detection.

The remainder of this paper is organized as follows. Section II reviews previous studies on object detection and image-sensor networks as an edge service. Section III presents the proposed scheme. Section IV compares ML models for object detection in terms of accuracy and computation time. Section V presents the results of a performance evaluation using a point-cloud dataset and demonstrates an implementation of the proposed scheme on an edge server. Section VI concludes this paper.

II. RELATED WORK

A. 3D-IMAGE SENSOR NETWORK

3D sensors are useful for controlling autonomous vehicles [8]. A number of networks composed of numerous 3D sensors have been deployed experimentally to help control autonomous vehicles. In 2006, the Intelligent Transportation Systems (ITS) group at the University of Minnesota developed a testbed system that placed a network of 3D sensors and radars near a rural intersection [9]. The network constantly monitored the intersection and collected road-traffic data from various perspectives. In 2009, Zhao et al. presented a system for monitoring an intersection using a network of 3D sensors and video cameras [10]. The expected output of this system consists of the motion trajectory of each moving object that entered the intersection and an estimation of the object's class, i.e., car, bus, bicycle, or pedestrian. The system performs data transformations within the horizontal plane, i.e., by using two translation parameters and one rotation parameter, between the coordinate systems of two neighboring 3D sensors. When a 3D sensor is used as the reference frame, all laser points from different 3D sensors are transformed into a common coordinate system. In addition, the client computers are connected through a network to a server computer, which broadcasts its local time periodically to synchronize the time of all computers. In 2011, Zhao *et al.* devised an algorithm for object detection and tracking using 3D sensors installed at intersections [11]. In 2014, Strigel et al. collected a dataset at a public intersection in Aschaffenburg, Germany [3]. In their study, four 3D sensors covered a wide area of the central intersection, two 3D sensors observed the sidewalks along the main road, and eight 3D sensors observed three egresses of the intersection. The 3D sensors synchronously operated at a frequency of 12.5 Hz. In 2019, Lv et al. devised a sensor infrastructure that actively senses surrounding traffic with roadside 3D sensors and broadcasts messages in real-time to connected vehicles via dedicated-short-range-communication (DSRC) roadside units [12]. Such broadcasted information may include position corrections, a local map, and safety messages [13], [14]. None of these studies addressed the problem of how to shorten the computation time of object detection by using ML on an edge server.

B. OBJECT DETECTION FROM POINT-CLOUD DATA

Deep learning has been used on point clouds for handling various tasks [5]. A well-studied task is detecting multiple objects in a point cloud of a scene. The methods of detection are categorized by the architecture of the detection process: region proposal and single shot [5].

Region-proposal methods have a two-stage architecture: first, regions where objects are likely to exist are identified. These regions are called proposals. Second, features are extracted from each proposal to determine its label. Region-proposal methods are further categorized as multi-view, segmentation, or frustum [5]. The multi-view type generates 3D rotated boxes by fusing the features from different view maps such as LIDAR front view, bird's eye view, and images. The segmentation type uses semantic segmentation techniques to remove most of the background points and generate large numbers of high-quality proposals on foreground points. PointRCNN, a segmentation type developed by Shi *et al.* [15], directly segments point clouds to obtain foreground points and then fuses semantic local-spatial features to produce high-quality 3D boxes. The frustum type uses 2D object detection techniques to generate 2D object proposals and makes 3D frustum proposals from each 2D proposal. A number of region-proposal methods, besides the multi-view, segmentation, and frustum types, have been devised. Namely, Shi et al. developed PointVoxel-RCNN (PV-RCNN) [16], which uses a 3D convolutional neural network and PointNet-based set abstraction [17] to extract features of a point cloud. The input point cloud is voxelized and the result is fed into a 3D sparse convolutional neural network to generate 3D proposals. The learned voxel-wise features are then encoded into a small set of key points via the voxel set abstraction module. Finally, key points are aggregated into region of interest (RoI) grid points. The features for each proposal are learned and refined. A confidence level is given to each proposal. Shi et al. also developed PART-A^2 [18], a neural network that consists of two stages: an object-part-aware stage and an object-part aggregation stage. The part-aware stage applies sparse convolution and sparse deconvolution to the input point cloud to predict intra-object part locations and generates 3D proposals. The part-aggregation stage conducts RoI-aware point cloud pooling to aggregate the object part information for each 3D proposal. The object part aggregation network is applied to score boxes and it refines the locations of objects on the basis of the features of the parts and information from the part-aware stage.

In contrast to the region-proposal-based methods, single-shot methods generate 3D bounding boxes directly through a single-stage network without generating proposals or post-processing. Single-shot methods are categorized on the basis of the type of input data: BEV-based, discretization-based, or point-based [5]. The BEV-based type takes a bird's eye view (BEV) representation as input, while the point-based type takes raw points as input. The discretization-based type–examples of which are SECOND [19] and PointPillars [20]–converts a point cloud into a regular discrete representation. In particular, SECOND takes raw points as input and uses VoxelNet [21] to divide the point cloud into evenly spaced voxels and encodes the features within each voxel into a 4D tensor. Then, by using a sparse convolution network [19], [22], the region-proposal network (RPN) generates object-detection results. PointPillars [20] organizes the input point cloud into columns (pillars) by using a stacked pillar tensor and a pillar index tensor. It uses PointNet [17] to learn the features of the point cloud that has been organized into pillars and encodes the learned features as a pseudo image. A 2D object-detection pipeline is then used to determine 3D bounding boxes.

There are two methods of generating proposals and bounding boxes: anchor-base and anchor-free [18]. The anchor-base method defines anchors, i.e., the average box size for each object, in advance and applies them sequentially to the feature map. The anchor-free method generates bounding boxes directly from points belonging to the target objects by regressing the difference of each point from the center. Point-RCNN uses the anchor-free method, while SECOND, Pointpillars, and PV-RCNN use the anchor-base method. PART-A^2 uses both methods.

C. REAL TIME OBJECT DETECTION AS EDGE SERVICES

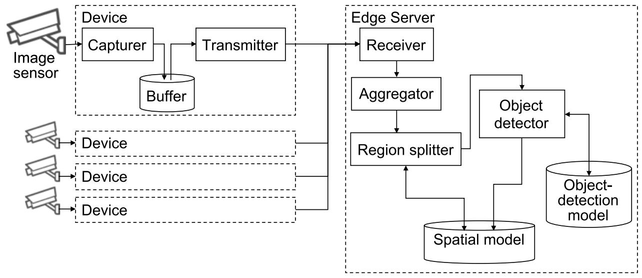
Industry investment and research into edge computing, in which computing and storing capabilities are placed at the ''Internet's edge'' in close proximity to mobile devices or sensors, have grown dramatically in recent years. Edge computing enables highly responsive cloud services for supporting mobile computing, scalability, and privacy-policy enforcement for the Internet of Things (IoT) and masking cloud outages [23]. Edge computers are expected to use machine learning, in particular, deep learning, so that they can obtain effective representations of data in real time. However, due to the limited computing capabilities and energy budgets of edge computers, it is difficult for them to analyze data used in real-time applications [24]. Here, Seyed Yahya *et al.* proposed a lightweight convolutional neural network (L-CNN) to handle the problem of using resource-limited edge computers for object detection [6]. Depthwise separable convolution is used to reduce the computing cost of a CNN without sacrificing its accuracy [25]. The number of filters is also reduced by focusing on human detection. A single-shot multi-object detector (SSD) [26] is fused to the CNN architecture to detect humans quickly and reliably. Seyed Yahya *et al.* also presented a framework for transferring knowledge between a deep neural network (DNN) and a shallow neural network (SHNN) [27]. The SHNN helps to suppress the energy consumed by the user-end device and the edge devices. The edge device uses the DNN, not for detecting objects, but for detecting changes in the image and updating the SHNN model, because the diversity of objects appearing in any image over a period of time is limited [28]. In particular, new knowledge (such as ground truth encoded with new weights) is transferred from each edge device to the user-end device to update the SHNN model. Compared with a framework that uses only a DNN model, Seyed Yahya *et al.*'s framework reduces inference time and energy consumption while keeping the loss in accuracy at an acceptable level.

III. PROPOSED SCHEME

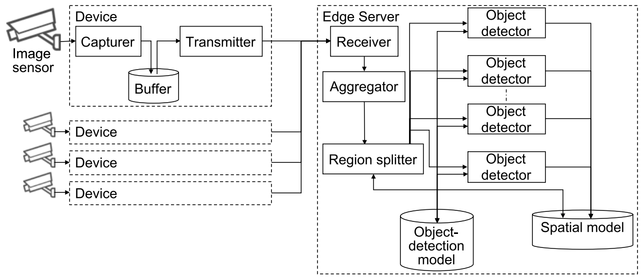
A. SYSTEM MODEL

The system models of the proposed scheme are illustrated in Fig. 1. One model (a) is for the sequential method; the other is for the parallel method (b). The system consists of an edge server and multiple devices equipped with image-sensor units (hereafter called sensors). Each sensor generates a stream of point-cloud data frame by frame, i.e., sequentially. The point-cloud data are transmitted from the sensors to the edge server via a communication channel.

Each frame, the image capturer receives point-cloud data from its corresponding image sensor unit and stores them in the buffer. The transmitter transmits the data stored in the buffer to the receiver of the edge server. The aggregator aligns (transforms) the coordinates of the point-cloud data received from each sensor and it aggregates the transformed point-cloud data frame by frame. The region splitter splits the



(a) System model used in the sequential method



(b) System model used in the parallel method

FIGURE 1. System models used in the proposed scheme.

point-cloud data into regions on the basis of the spatial importance determined from the spatial model. The object detector detects pedestrians, cars, and bicycles from the point-cloud data of each frame by using the pre-trained ML model on the cloud server. The sequential method performs object detection sequentially region by region, while the parallel method performs object detection on multiple regions in parallel by using multiple object detectors.

Fig. 2 presents an example of how the sequential method works. First, a point cloud is divided into regions in accordance with their spatial importance scores; regions with higher scores more likely contain objects such as cars, trucks, bicycles, and pedestrians. Then, object detection is performed on each region sequentially in descending order of the score of the region. Fig. 2b illustrates the parallel method. After the point cloud is divided into regions, object detection is performed on every region in parallel. Since the regions with low spatial importance scores contain a lot of background such as roads and buildings, it is expected that the computation time for the regions with higher scores of spatial importance should be shorter.

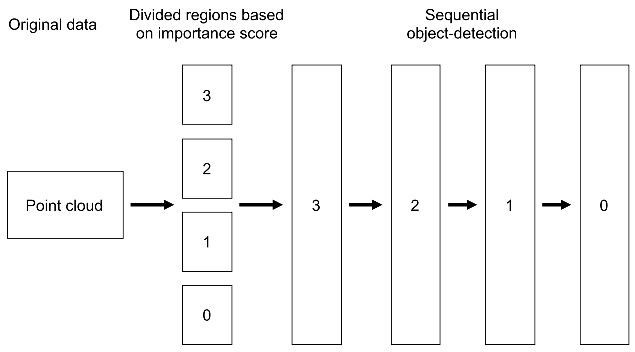
B. PROBLEM FORMULATION

The problem of minimizing the total computational time of object detection for the sequential method is as follows.

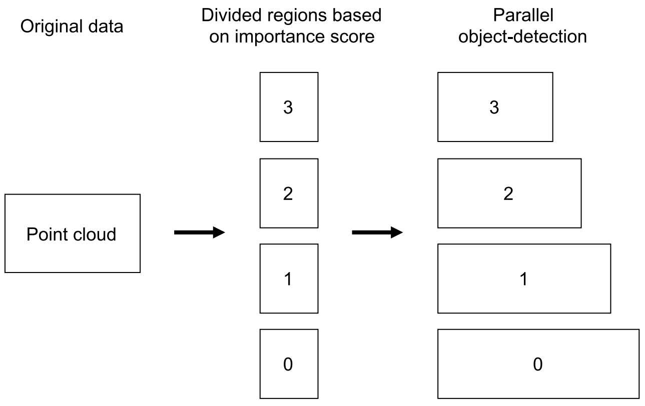
$$\min \sum_{i\in I} t_s(d(x_{i(r)})), \tag{1}$$

$$\text{s.t.} \sum_{i\in I} a(d(x_{i(r)})) > \alpha. \tag{2}$$

Here, $t_s()$ denotes the time duration from the start to the end of the computation, which equals the total computation time using a model for object detection in the regions of point-cloud data. A set of all divided regions I and the


Original data
Divided regions based on importance score
Sequential object-detection
Point cloud
3
2
1
0
3
2
1
0

(a) Example of sequential processing in the proposed scheme


Original data
Divided regions based on importance score
Parallel object-detection
Point cloud
3
2
1
0
3
2
1
0

(b) Example of parallel processing in the proposed scheme

FIGURE 2. Example of prioritized processing in the proposed scheme.

required accuracy of object detection α are given parameters. The division format r, which determines how to split the point cloud into multiple regions in accordance with the spatial importance, is a decision variable. $x_{i(r)}$ denotes the point-cloud data of region $i \in I$ produced using r. $d()$ denotes the object detection model with $x_{i(r)}$ as its input. $a()$ denotes the accuracy, such as the precision and recall, of object detection on the divided point cloud. Equation (1) is the objective of the problem, which is to minimize the total computational time of object detection on the divided point-cloud data by appropriately selecting r. Equation (2) is the constraint of the problem wherein the overall accuracy of object detection in the regions must be greater than α.

As for the parallel method, the problem is to minimize the longest of the computational times on the divided point-cloud data by appropriately selecting r:

$$\min \max_{i\in I}(t_p(d(x_{i(r)}))), \tag{3}$$

$$\text{s.t.} \sum_{i\in I} a(d(x_{i(r)})) > \alpha. \tag{4}$$

Here, $t_p()$ denotes the time duration from the start to the end of the computation, which equals the longest computation time for a region. Equation (3) is the objective of the problem, while Equation (4), the constraint, is identical to (2).

Algorithm 1 Control Procedure of Proposed Scheme

```
    Model-creation phase
1:  $f_1, f_2, \dots, f_l$ ← $l$ frames of point cloud collected in advance
2:  $P$ ← spatial importance estimated from aggregation of $f_1, f_2, \dots, f_l$
    Real-time object detection phase
3:  $R_j (j = 1, 2, \dots, m)$ ← regions determined by spatial importance $P$, where $m$ is the number of regions; $R_j$ with smaller $j$ has higher spatial importance.
4:  $p_i (i = 1, 2, \dots, n)$ ← point-cloud data obtained by sensors in real time, where $n$ is the number of sensors.
5:  for $i = 1, 2, \dots, n$ do
6:      for $j = 1, 2, \dots, m$ do
7:          Pick a set of points in $R_j$ from $p_i$ and add them to $q_j$
8:      end for
9:  end for
10: if $S$ is True (Sequential method) then
11:     for $j = 1, 2, \dots, m$ do
12:         Run object detection for $q_j$ sequentially
13:     end for
14: else (Parallel method)
15:     Run object detection for $q_1, q_2, \dots, q_m$ in parallel
16: end if
```

C. CONTROL PROCEDURE

Algorithm 1 is the control procedure for prioritizing the computations of the proposed scheme. It roughly consists of two phases: model creation and real-time object detection. In the model-creation phase, the spatial importances are estimated. (The way of estimating the spatial importance is explained in Section V-A.) In the real-time object-detection phase, point-cloud data are divided up into regions on the basis of their spatial importance. In the sequential method, object detection is performed on the regions in descending order of their spatial importance. In the parallel method, object detection is performed for each divided region in parallel.

IV. COMPARISON OF OBJECT DETECTION METHODS

We examined a number of object-detection methods to find ones suitable for the evaluation of the proposed scheme.

A. DATASET

The KITTI dataset, which was published by Geiger *et al.* [29], has been extensively used in studies on mobile robotics and autonomous driving. The dataset contains 3D image data recorded while a car, fitted with a Velodyne HDL-64E LIDAR unit, was driven in and around Karlsruhe, Germany. The raw data, collected at various locations (from local highways to urban scenes), cover a wide variety of autonomous driving scenarios with static and dynamic objects. The cameras and LIDAR unit were installed 1.73 m and 1.65 m above the ground and synchronously operated at 10-100 Hz. The dataset contains raw LIDAR data, camera images, the calibration matrix of the camera, and object-label information, which are all calibrated, synchronized, and timestamped. Eight object labels "car," "van," "truck," "pedestrian," "person," "cyclist," "tram," and "misc." (e.g., trailers and Segways). The dataset is divided into training and test sets. The whole dataset contains 7,481 frames with label information. The average number of points per frame is around 120,000.

TABLE 1. Classification of object-detection methods.

Method	First step	Second step	Proposal
SECOND	Voxel-base	–	Anchor-base
PART-A^2	Voxel-base	Voxel-base	Anchor-base
PART-A^2-free	Voxel-base	Voxel-base	Anchor-free
PV-RCNN	Voxel-base	Point-base	Anchor-base

B. OBJECT DETECTION METHODS

The Open-source Toolbox for 3D Object Detection from Point Clouds (OpenPCDet), which was published by OpenPCDet Development Team [30], has been extensively used for object detection. We used this toolbox to implement the object-detection methods. Although OpenPCDet provides six object-detection methods, we compared only four of them: SECOND [19], PV-RCNN [16], PART-A^2-free [18], and PART-A^2 [18]. We used these methods with the default models; it is included in future work how to set parameters for the model of each method. We chose the "car," "pedestrian," and "cyclist" benchmarks from KITTI [31]. For each class, three levels of difficulty were set: "easy," "moderate," and "hard." We used 3,712 frames for training and 3,769 frames for validation. The detection results were evaluated in terms of computation time per frame, which equals to the total computation time divided by the total number of frames, and average precision (AP), which is used extensively as a metric for detection accuracy [31].

C. PERFORMANCE COMPARISON

The cloud server was an Amazon EC2 g3.4xlarge with Ubuntu OS version 18.04.5 equipped with sixteen Intel Xeon E5-2686 v4 CPUs @ 2.7GHz with 122GiB memory and one NVIDIA Tesla M60 GPU with 8GiB memory.

Fig. 3 shows the cumulative distribution function (CDF) of computation time per frame measured for each object detection method. SECOND is the fastest, which is reasonable because it is a single-shot method that provides bounding boxes directly without having to generate proposals or aggregate features, while the others are region-proposal-based methods. Table 2 shows the AP of each object-detection method on the KITTI benchmarks. SECOND had almost the same AP for cars as the other methods, although its APs for pedestrians and cyclists were inferior to those of the other methods. This comparison indicates that SECOND is reasonable choice for evaluating the performance of the proposed scheme. That is, we consider computation time to be the

TABLE 2. Average precision of each object-detection method.

Method	Cars			Pedestrians			Cyclists		
Difficulty	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
SECOND	90.6	81.6	78.6	55.9	51.1	46.2	83.0	66.7	62.8
PART-A^2	92.2	82.9	82.0	66.9	59.7	54.6	90.3	70.1	66.9
PART-A^2-free	91.7	80.3	78.1	72.4	66.40	60.1	91.9	75.3	70.6
PV-RCNN	92.1	84.4	82.5	62.7	54.5	49.9	89.1	70.4	66.0

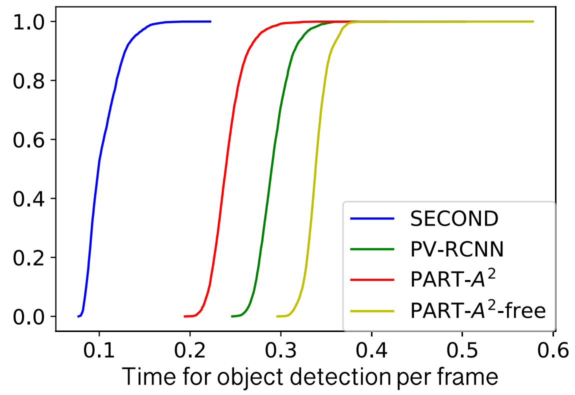


FIGURE 3. Cumulative distribution function of computational time of object detection per frame.

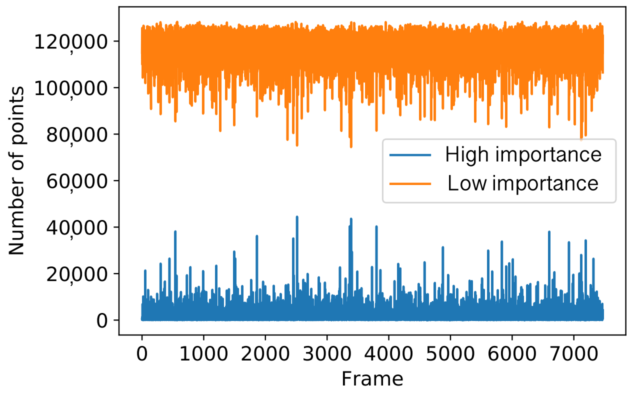


FIGURE 4. Number of points in high- and low-importance regions.

primary factor limiting the capability of edge computing, and SECOND is faster than the other methods, while its AP for cars is acceptable.

D. SECOND

SECOND [19] consists of four parts: a voxel features and coordinates converter, voxel feature extractor, sparse convolution layers, and RPN. It takes a raw point cloud of a scene as input and converts them to voxel features and coordinates. Then it uses two voxel-feature encoding layers [21] and a fully connected network to extract the voxel features. It applies a sparse convolution [19], [22] network to the extracted features. The result of these operations converts the point cloud into a number of sparse voxels, 5,000 to 8,000 voxels with a sparsity of nearly 0.005 in case of KITTI [29]. Finally, the RPN performs the object detection.

V. PERFORMANCE EVALUATION

A. SPATIAL IMPORTANCE CLASSIFICATION

Before describing the results of the performance evaluation, let us describe how the point clouds were divided into importance classes. Regions containing bounding boxes extracted from the object-label information were assumed to be important in object detection because objects are likely to exist in such regions. For each object with a bounding box, fifteen features are available from the object label information of the KITTI dataset. Seven of those features, namely, x position [m], y position [m], z position [m], object width [m], object length [m], object height [m], and rotation angle [rad], are used to extract the bounding boxes of the objects; the locations and rotation angles are defined in the camera coordinates. Then, the point cloud was divided into just two regions, i.e., a high- importance region and low-importance one, depending on whether the points were inside or outside the bounding boxes. Fig. 4 shows the numbers of points in the high- and low-importance regions. The low-importance region has twice as many point as the high-importance region.

TABLE 3. Computation time (in seconds) for all frames when running each method on a cloud server.

Method	High importance	Low importance	Overall
Sequential	265.5	865.1	1130.6
Parallel	441.9	893.3	893.3
Benchmark	-	-	848.8

TABLE 4. Computation time (in milliseconds) per frame when running each method on a cloud server.

Method	High importance	Low importance	Overall
Sequential	70.5	229.7	150.1
Parallel	117.3	237.2	237.2
Benchmark	-	-	225.4

B. EVALUATION RESULTS

The measurement environment was the computational environment described in Section IV-C. Frames with no points in the high-importance region were removed from the validation set. We compared the computation times, APs, and recalls of the proposed scheme and the benchmark scheme, SECOND, selected without splitting the data.

Table 3 lists the computation times for processing all frames of the data. The overall region means the whole cloud. The computation time for one frame can be derived

TABLE 5. Recall for each region.

	Cars			Pedestrians			Cyclists		
Region	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
High importance	87.7	76.0	72.2	79.6	73.5	68.1	86.5	67.5	62.8
Low importance	70.6	57.6	53.4	1.6	1.5	1.4	21.4	14.7	14.4
Overall	91.5	86.4	83.9	78.3	71.7	65.7	95.4	82.5	77.9

TABLE 6. AP for each region.

	Cars			Pedestrians			Cyclists		
Region	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
High importance	83.3	72.2	67.0	67.2	63.5	58.9	82.7	64.1	61.1
Low importance	59.0	44.4	40.6	0.0	0.0	0.0	2.4	1.2	1.1
Overall	90.6	81.6	78.6	55.9	51.1	46.2	83.0	66.7	62.8

by dividing the values by 3,766, except for "the sequential method for the overall region" where it can be derived by dividing the value by 7,532. The computation time for one frame of the high-importance region is shorter than those of the low-importance and overall regions. Object detection for the high-importance region took a shorter time because the number of points in the high-importance region was smaller than those in the low-importance and overall region. The benchmark method was faster at processing the points in the overall region. However, the sequential and parallel methods were faster on the high-importance region. Moreover, the computation time of the parallel method for the overall region was not much longer than that of the benchmark method. These results show that, by dividing the point-cloud data on the basis of spatial importance, the proposed scheme performed well in terms of reducing the computation time of object detection.

Tables 5 and 6 show the recalls and APs. In this table, the listed APs and recalls for the high- and low-importance regions are for the parallel and sequential methods, which performed identically in these terms. The "Overall" row indicates the results of the benchmark method, which does not split up the point-cloud data. Compared with the recall for the overall region, the recall for pedestrians in the high-importance region was high, while those for cars and cyclists in the high-importance region was low. The same trend appears for AP shown in Table 2. AP depends on the preciseness of the size and angle of the bounding boxes given by the label information in the dataset. The recalls for cars and cyclists in the low-importance region are not zero; namely, objects are detected even from points in the low-importance region. We analyzed the positions within the bounding boxes in the low-importance region and found out that some of the points of cars and cyclists (such as those at the front edges and rear edges of vehicles) were included in the low-importance region. Consequently, in the performance evaluation, if the label information was perfect, the accuracy of the high-importance region in terms of recall and AP would be identical with that of the benchmark.

TABLE 7. Computation time (in seconds) for all frames when running each method on an edge server. Values in brackets indicate ratios relative to the computation times listed in Table 3 when run on the cloud server.

Method	High importance	Low importance	Overall
Sequential	1399.6 (5.3)	2547.2 (2.9)	3946.8 (3.5)
Parallel	2350.5 (5.3)	5467.2 (6.1)	5467.2 (6.1)
Benchmark	–	–	2636.5 (3.1)

TABLE 8. Computation time in milliseconds per frame when running each method on an edge server. Values in brackets indicate ratios relative to the computation times in Table 4 when run on the cloud server.

Method	High importance	Low importance	Overall
Sequential	371.6 (5.3)	676.4 (2.9)	524 (3.5)
Parallel	624.1 (5.3)	1451.7 (6.1)	1451.7 (6.1)
Benchmark	–	–	700.1 (3.1)

C. IMPLEMENTATION OF PROPOSED SCHEME ON JETSON AS EDGE SEVER

The proposed scheme was implemented on an edge sever that had a much lower computation power than that of the cloud server used in the comparison described in Section IV-C. The edge server was a Jetson Xavier NX from NVIDIA. It is equipped with 384 NVIDIA CUDA®cores, 48 tensor cores, six Carmel ARM CPUs, and two NVIDIA deep-learning accelerators (NVDLA) engines. In an experiment with over 59.7 GB/s of memory bandwidth, video encoding and decoding, a Jetson Xavier NX was used to process high-resolution data from multiple sensors and to run multiple modern neural networks in parallel simultaneously [32]. Table 7 shows the computation times for processing all frames with the Jetson Xavier NX. The computation time per frame is also shown. The computation time of the edge server is 3.5 to 6.1 times longer than that of the cloud server shown in Table 3. This result was as expected because the computation power of the edge server is much lower than that of the cloud server. The computation time of the sequential and parallel methods used for the high-importance region was shorter than that of the benchmark method; this result suggests that the proposed scheme obtains object-detection results for the high-importance region quicker than the benchmark method.

VI. CONCLUSION

A spatial-importance-based computation scheme that works on edge computers of image-sensor networks using multiple LIDAR sensors was proposed. The scheme performs object detection sequentially or in parallel in accordance with the spatial importance of point-cloud data; it shortens the delay involved in object detection. Multiple object-detection methods were compared in a numerical study using the KITTI dataset. The point-cloud data of the dataset were divided into high- and low-importance regions, in which the points in each region were inside or outside bounding boxes. From the results of the study, we chose to compare our method with SECOND, a discretization type of detection method, in terms of accuracy and computation time (average precision and recall). The results of the evaluation showed that the proposed scheme with the sequential or parallel method reduces the computation time of object detection on cloud and edge servers.

ACKNOWLEDGMENT

The evaluation results were partly obtained from research commissioned by the National Institute of Information and Communications Technology (NICT), Japan.

REFERENCES

[1] G. P. Hancke, B. de Carvalho e Silva, and G. P. Hancke, "The role of advanced sensing in smart cities," *Sensors*, vol. 13, no. 1, pp. 393–425, 2012. [Online]. Available: https://www.mdpi.com/1424-8220/13/1/393

[2] S. R. E. Datondji, Y. Dupuis, P. Subirats, and P. Vasseur, "A survey of vision-based traffic monitoring of road intersections," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 10, pp. 2681–2698, Oct. 2016.

[3] E. Strigel, D. Meissner, F. Seeliger, B. Wilking, and K. Dietmayer, "The ko-PER intersection laserscanner and video dataset," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 1900–1901.

[4] K. Sato, R. Shinkuma, T. Sato, E. Oki, T. Iwai, D. Kanetomo, and K. Satoda, "Prioritized transmission control of point cloud data obtained by LIDAR devices," *IEEE Access*, vol. 8, pp. 113779–113789, 2020.

[5] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3D point clouds: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 12, pp. 4338–4364, Dec. 2021.

[6] S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B.-Y. Choi, and T. R. Faughnan, "Real-time human detection as an edge service enabled by a lightweight CNN," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, Jul. 2018, pp. 125–129.

[7] R. Otsu, R. Shinkuma, T. Sato, and E. Oki, "Data-Importance-Aware bandwidth-allocation scheme for point-cloud transmission in multiple LIDAR sensors," *IEEE Access*, vol. 9, pp. 65150–65161, 2021.

[8] R. Dominguez, E. Onieva, J. Alonso, J. Villagra, and C. Gonzalez, "LIDAR based perception solution for autonomous vehicles," in *Proc. 11th Int. Conf. Intell. Syst. Design Appl.*, Nov. 2011, pp. 790–795.

[9] L. Alexander, P.-M. Cheng, A. Gorjestani, A. Menon, B. Newstrom, C. Shankwitz, and M. Donath, "The Minnesota mobile intersection surveillance system," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Sep. 2006, pp. 139–144.

[10] H. Zhao, J. Cui, H. Zha, K. Katabira, X. Shao, and R. Shibasaki, "Sensing an intersection using a network of laser scanners and video cameras," *IEEE Intell. Transp. Syst. Mag.*, vol. 1, no. 2, pp. 31–37, Sep. 2009.

[11] H. Zhao, J. Sha, Y. Zhao, J. Xi, J. Cui, H. Zha, and R. Shibasaki, "Detection and tracking of moving objects at intersections using a network of laser scanners," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 655–670, Jun. 2012.

[12] B. Lv, H. Xu, J. Wu, Y. Tian, Y. Zhang, Y. Zheng, C. Yuan, and S. Tian, "LiDAR-enhanced connected infrastructures sensing and broadcasting high-resolution traffic information serving smart cities," *IEEE Access*, vol. 7, pp. 79895–79907, 2019.

[13] S. Atev, H. Arumugam, O. Masoud, R. Janardan, and N. P. Papanikolopoulos, "A vision-based approach to collision prediction at traffic intersections," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 4, pp. 416–423, Dec. 2005.

[14] Z. Tian, P. Michael Kyte PHD, and H. Liu, "Vehicle tracking and speed measurement at intersections using video detection systems," *Inst. Transp. Eng. ITE J.*, vol. 79, no. 1, p. 42, 2009.

[15] 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.

[16] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li, "PV-RCNN: Point-voxel feature set abstraction for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10529–10538.

[17] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 652–660.

[18] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3D object detection from point cloud with part-aware and part-aggregation network," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 8, pp. 2647–2664, Aug. 2020.

[19] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[20] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12697–12705.

[21] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4490–4499.

[22] B. Graham and L. van der Maaten, "Submanifold sparse convolutional networks," 2017, *arXiv:1706.01307*.

[23] M. Satyanarayanan, "The emergence of edge computing," *Comput.*, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[24] G. Plastiras, M. Terzi, C. Kyrkou, and T. Theocharidcs, "Edge intelligence: Challenges and opportunities of near-sensor machine learning applications," in *Proc. IEEE 29th Int. Conf. Appl.-Specific Syst., Archit. Processors (ASAP)*, Jul. 2018, pp. 1–7.

[25] L. Sifre and S. Mallat, "Rigid-motion scattering for image classification," Ph.D. dissertation, Ecole Polytechnique, France, 2014.

[26] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 21–37.

[27] S. Y. Nikouei, Y. Chen, S. Song, and T. R. Faughnan, "Kerman: A hybrid lightweight tracking algorithm to enable smart surveillance as an edge service," in *Proc. 16th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2019, pp. 1–6.

[28] M. Farhadi and Y. Yang, "TKD: Temporal knowledge distillation for active perception," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 953–962.

[29] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.

[30] O. D. Team. *Openpcdet: An Open-Source Toolbox for 3D Object Detection From Point Clouds*. Accessed: Nov. 5, 2021. [Online]. Available: https://github.com/open-mmlab/OpenPCDet

[31] *Kitti 3D Object Detection Benchmark Leader Board*. Accessed: Nov. 5, 2021. [Online]. Available: http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d

[32] NVIDIA. *Jetson Xavier NX Developer Kit | Nvidia Developer*. Accessed: Nov. 5, 2021, [Online]. Available: https://developer.nvidia.com/embedded/jetson-xavier-nx-devkit

RYO OTSU received the B.E. degree in electrical and electronic engineering from Kyoto University, in 2019, and the master's degree from the Graduate School of Informatics, Kyoto University, in 2021. His research interest includes 3D-image sensor networks.

RYOICHI SHINKUMA (Senior Member, IEEE) received the B.E., M.E., and Ph.D. degrees in communications engineering from Osaka University, in 2000, 2001, and 2003, respectively. He joined the Graduate School of Informatics, Kyoto University, and worked there as an Assistant Professor, from 2003 to 2011, and as an Associate Professor, from 2011 to 2021. He was a Visiting Scholar at the Wireless Information Network Laboratory, Rutgers University, from 2008 to 2009. In 2021, he joined the Faculty of Engineering, Shibaura Institute of Technology, as a Professor. His main research interest includes cooperation in heterogeneous networks. He is a fellow of the IEICE. He received the Young Researchers' Award from the IEICE in 2006, the Young Scientist Award from Ericsson Japan in 2007, the TELECOM System Technology Award from the Telecommunications Advancement Foundation in 2016, and the Best Tutorial Paper Award from the IEICE Communications Society in 2019. He was the Chairperson of the Mobile Network and Applications Technical Committee of the IEICE Communications Society, from 2017 to 2019.

TAKEHIRO SATO (Member, IEEE) received the B.E., M.E., and Ph.D. degrees in engineering from Keio University, in 2010, 2011, and 2016, respectively. From 2011 to 2012, he was a Research Assistant with the Keio University Global COE Program, "High-Level Global Cooperation for Leading-Edge Platform on Access Spaces," established by the Ministry of Education, Culture, Sports, Science and Technology of Japan. From 2012 to 2015, he was a Research Fellow with the Japan Society for the Promotion of Science. From 2016 to 2017, he was a Research Associate at the Graduate School of Science and Technology, Keio University. He is currently an Assistant Professor at the Graduate School of Informatics, Kyoto University. His research interests include communication protocols and network architecture for next-generation optical networks. He is a member of the IEICE.

EIJI OKI (Fellow, IEEE) received the B.E. and M.E. degrees in instrumentation engineering and the Ph.D. degree in electrical engineering from Keio University, Yokohama, Japan, in 1991, 1993, and 1999, respectively. He was with Nippon Telegraph and Telephone Corporation (NTT) Laboratories, Tokyo, from 1993 to 2008, and the University of Electro-Communications, Tokyo, from 2008 to 2017. From 2000 to 2001, he was a Visiting Scholar at Polytechnic University, Brooklyn, NY, USA. In 2017, he joined Kyoto University, Japan, where he is currently a Professor. His research interests include routing, switching, protocols, optimization, and traffic engineering in communication and information networks. He is a fellow of the IEICE.

DAIKI HASEGAWA received the B.E. and M.E. degrees in civil engineering from Kyoto University, in 2009 and 2011, respectively. He is currently with the development of AI-enabled services for industrial innovation and social problems solutions, ExaWizards Inc.

TOSHIKAZU FURUYA received the B.E. degree from the Faculty of Culture and Information Science, Doshisha University, in 2010, and the master's degree in business administration from Kyoto University, in 2014, where he is currently pursuing the Ph.D. degree with the Graduate School of Informatics. He founded ExaWizards Inc., as an AI startup company, in 2016. He is currently the CEO of Quantum Analytics, Inc.