Object Recognition from 3D Point Cloud on Resource-Constrained Edge Device

Yuma Okochi*, Hamada Rizk*†, Tatsuya Amano*, Hirozumi Yamaguchi*
*Graduate School of Information Science and Technology, Osaka University, Japan, † Tanta University, Egypt
*{y-okochi, hamada_rizk, t-amano, h-yamagu}@ist.osaka-u.ac.jp

***Abstract*—This paper presents the design and development of a lightweight, portable 3D spatial sensing device equipped with a compact LiDAR-type sensor. The device provides a 3D point cloud representation of the surrounding environment as captured by the sensor. Based on the acquired 3D point cloud, we propose a real-time object recognition method. The technical challenge is how to process the 3D point data in real-time while pursuing the best trade-off between the processing overhead and recognition accuracy. To answer the question, we leverage the Fisher Vector to extract spatio-temporal features of different objects enabling efficient classification of these objects using the Support Vector Machine approach. The experimental results show that the proposed method achieves mean Average Precision of 0.961. The processing speed on the device was 59.3 frames/second, indicating that object detection can be done in real-time on the device.**

***Index Terms*—3D Point Cloud, Object Recognition, Spatial Sensing, Edge Device**

I. Introduction

The recent advancements in mobile IoT unleash an extraordinary cycle of innovative applications that target humans, vehicles, and mobile objects. These applications build a secure and enriched living environment. They will also lead to the realization of the next generation of human-centered intelligent environments that will improve the human quality of life. Particularly, the human sensing field is witnessing a growing need for applications that support social distancing and contact tracing to prevent the spread of infectious diseases (e.g., COVID-19) [1]–[4]. These applications also contribute to understanding and analyzing human flow in working environments, or commercial facilities [5], as well as in-home monitoring of the elderly for emergency detecting (e.g., falls).

Most of these detection systems use image analysis technology based on RGB cameras [6], which is not an acceptable solution in private environments. This is because of the leakage of personal information (e.g., individuals' faces) and privacy-sensitive information (e.g., appearance and clothing). Even in public spaces such as commercial facilities, introducing this technology as data acquisition means without consent may not be allowed, and an opt-out mechanism is required in some cases.

To solve this problem, our research group is developing a human flow detection system called "Hitonavi" [7] that uses 3D point clouds acquired by Light Detection And Ranging (LiDAR) sensors. Since the 3D point cloud acquired by LiDARs does not have color information, there is little risk of invading subjects' privacy. Powerful, high-precision LiDARs can precisely capture the shape of subjects in a wide area, and it has been used for object detection in indoor and outdoor environments [8] and person tracking [9], [10]. However, since these LiDAR systems acquire dense point clouds with large volume, data processing needs high-performance computational resources.

In this paper, we present the design and development of a small, portable device, called *Hitonavi-*μ, which is equipped with an ultra-compact and power-efficient LiDAR-type device and a microcontroller board, to acquire a 3D point cloud of the surrounding environment. We also propose a real-time object recognition method using the 3D point cloud obtained by the device. The method is lightweight enough to run on the device, which enables real-time processing of the obtained 3D point cloud with good segmentation and classification accuracy of the objects in the scene. The key idea is to leverage the Fisher Vector representation of the 3D point cloud features and a trained SVM classifier tailored to the object recognition.

Hitonavi-μ can be used for supporting visually impaired pedestrian while wearing the device [11]. The device can recognize the surrounding objects and alert risks of collisions [12]. If the device is on a worker's desk, it can detect work posture and assess her/his concentration level during the work [13].

The evaluation results on a dataset using the *Hitonavi-*μ device show that the proposed method achieves 0.961 mAP. This indicates that the features of the point cloud distribution can recognize the objects and their types with sufficiently high accuracy. Besides, the processing rate on the device was 59.3 frames/second. This highlights the real-time performance of the proposed method on resource-constrained edge devices, especially off-the-shelf microcomputers without GPUs.

II. Related Work

Object recognition in a 3D point cloud is generally defined as a segmentation and classification problem to find objects in the scene and determine the classes (types) of objects. For this purpose, many efforts have been dedicated so far [14]–[17]. PointNet [15] is a well-known approach. Given a 3D point cloud as input, a multilayer perceptron extracts features from the point cloud and performs segmentation of the point cloud. The method applies Max-pooling, a symmetric function that preserves the point cloud's sequential invariance. PointNet++ [16] is an enhanced version of PointNet that improves accuracy by incorporating information from each

978-1-6654-6975-3/22/$31.00 ©2022 IEEE

point's neighbors. VoteNet [17] focuses on the fact that point clouds acquired by geodesic sensors capture only the surface of objects and uses a voting mechanism to determine the center point of objects. These methods enable highly accurate object recognition. However, they assume powerful hardware for both sensing and data processing. More specifically, these methods assume point clouds with high density obtained by powerful LiDARs or depth cameras, which consume much energy for sensing. They also rely on deep learning-based data processing, which is computationally expensive and often requires high-performance GPUs for training and inference. Therefore, running these methods on resource-constrained edge devices is not straightforward.

Meanwhile, in this study, we explore a possibility of object recognition using our portable device with a small LiDAR-like sensor and an off-the-shelf microcomputer. The obtained point cloud is much more sparse than that from such powerful sensing devices. Therefore, it is not easy to capture the features of the objects, achieving sufficient accuracy and processing speed. To pursue the best balance, we leverage a combination of a pillar grid-based discrimination for fast segmentation and a compact representation of point distribution using Fisher Vectors (FVs) and a conventional classifier (SVM) for class inference.

Several methods have been proposed for processing 3D point clouds using mobile devices. Kim et al. [18] employed a hardware-dependent approach using a graph convolutional network with pipelining, and achieved high-speed, energy-efficient point cloud processing. However, they need dedicated hardware (the processor architecture has been optimized for the proposed processing scheme), and the algorithm has been configured for the architecture. Meanwhile, our goal is to develop a cost-efficient device with a small single-board computer (we use a raspberry pi 4 compute module) that has an ARM-compatible CPU and an algorithm that works on it. Liu et al. [19] propose a LiDAR-equipped mobility support system for visually impaired people. A wearer is equipped with a LiDAR and a laptop PC on her/his body and back, respectively. The system can generate a 3D map of the environment in front using SLAM, perform object recognition, capture the surrounding environment, and provide voice notification. Although the system has shown the possibility of a pedestrian SLAM system using LiDARs, it needs powerful machines with GPUs and sensing devices, which differs from our approach that only relies on a simple, cost-efficient device with a small LiDAR.

To summarize, the existing approaches to enable point cloud processing on edge devices do not satisfy the following requirements that we think are significant for our purpose. (i) The data processing should be lightweight and hardware-independent, which can easily be implemented and run on off-the-shelf CPU-based architecture, and (ii) the point cloud data can be sparse to allow energy-efficient small sensing devices.

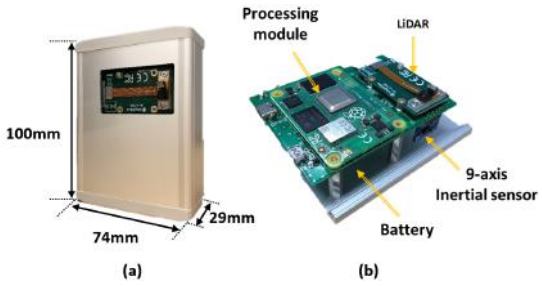

Fig. 1. Device dimensions and components

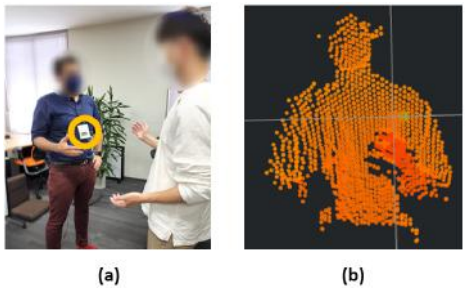

Fig. 2. (a) Device usage, (b) acquired point clouds

III. Device Architecture

In this section, we describe the architecture of *Hitonavi-μ*. *Hitonavi-μ* can obtain a 3D point cloud (frame) of the surrounding[1]. It consists of a battery, a microcontroller unit, and a MEMS LiDAR sensor, as shown in Fig 1. The components are assembled into a case of $100mm(H)\times74mm(W)\times29mm$ (D) made of aluminum and plastic. The case has a hole for the LiDAR sensor, a type-C port for battery charging, and a power switch. The processing unit of *Hitonavi-μ* is a Raspberry Pi Compute Module 4 (CM4). This unit is connected to an I/O board that provides the required interfaces to the LiDAR sensor and power supply. The CM4 has an ARM Cortex-A72 CPU (1.50 GHz, four cores) with 4 GB of RAM, which facilitates running our algorithms on the edge.

The LiDAR sensor has an ultra-compact size with a significantly lighter weight than existing LiDAR devices in the market. The sensor consists of an infrared laser and a CMOS image sensor and can acquire a 3D point cloud in real-time. Additionally, the detection range of this sensor is within 3m, and the FOV (Field Of View) is 90 degrees.

Hitonavi-μ is also equipped with a 9-axis inertial measurement unit of tri-axial accelerometer, gyroscope, and magnetometer, whose operating frequency ranges from 14.9 Hz to 952 Hz. The acquisition of acceleration, angular velocity, and magnetic measurements enables estimating the user's posture wearing the device. In addition, the device is equipped with a battery to supply power when used as a portable device.

The wearing usage of *Hitonavi-μ* and the acquired point cloud are shown in Fig. 2. The device is worn around the neck as shown in (a) and acquires the shape of objects within

[1]Each point in the cloud has only x, y, and z coordinate information without color or other information.

the detection range in the form of a 3-D point cloud as shown in (b). That point cloud can represent the shape of a person in fine detail.

IV. Object Recognition Algorithm

Fig. 3 shows the proposed 3D point cloud segmentation and classification procedure. The proposed method takes the 3D point cloud obtained from the LiDAR as input. First, the method divides the input point cloud by *pillar-grid* and performs grid-based clustering process to segment the point clouds into objects (Section IV-A). Fisher Vector-based features are extracted from each cluster to determine the shape of the objects (Section IV-B). The obtained feature vectors are used as input to the machine learning model (SVM) to learn object class inference, and a class label is assigned to each point according to the inference results (Section IV-C). Detailed descriptions of these sub-processes are given in the following subsections.

A. Pillar-grid-based Segmentation

PointPillars [20], an object detection method, uses a pillar grid to divide the point cloud into pillars to speed up processing. We leverage the concept of that approach.

Let $L = \{\boldsymbol{p}_i \in \mathbb{R}^3,\ i = 1, ..., n\}$ be the set of 3D points, where n denotes the number of points in a given point cloud. Each point has only geometric information in 3D space (x_i, y_i, z_i). First, the point cloud is discretized into an evenly spaced grid in the x-z plane, creating a set $\mathcal{P}$ of pillars.

Next, a grid-based clustering process is performed. Fig. 4 illustrates the process. Before clustering, noise reduction is performed. Using the number of points in each grid, if the number of points is above a threshold, the grid is considered as a component to form an object (the green grids in Fig. 4). If the number of points is less than the threshold, the grid is considered as noise (gray grid in Fig. 4) and is not included in the subsequent clustering process.

Finally, a grid-based clustering process is performed. If both adjacent grids are object-presence grids (green grids), they are considered to be in a single cluster. By repeating this process, the cluster can be divided into three clusters in this example, as shown in Fig. 4.

The time complexity of this processes is $O(n)$, which is faster than clustering methods such as DBSCAN [21] (worst-case complexity $O(n^2)$), which are often used in object detection.

We note that, if the pillar-grid size is too small, it may be difficult to determine if the point cloud in each pillar is from an object or noise, resulting in lower detection accuracy. Meanwhile, with too large-sized pillars, a single grid may contain point clouds from two or more objects, decreasing recognition accuracy. In our implementation, we determine the grid size as $20cm \times 20cm$ empirically.

B. Feature Extraction via Fisher Vector

Next, we extract features based on each selected pillar's Fisher Vector (FV) representation. Generally, processing point cloud data has a lot of challenges. Extracting meaningful context is not straightforward due to its unordered, unstructured, and varying size nature, unlike visible images. FV representation can overcome these challenges because it is independent of their order and sample size. Our method uses feature vectors from point cloud distribution in each cluster (*i.e.*, each segment found in the previous section). FV representation is employed to define discriminative spatio-temporal signatures for different objects' data of variable sizes. The intuition behind using FV for feature extraction is its ability to capture the spatial formation of 3D points in space, yielding discriminative signatures of the objects. These signatures are defined as the deviation of 3D points from the Gaussian Mixture Model (GMM). This can be done by calculating the gradients of the sample's log-likelihood w.r.t. the model parameters (*i.e.*, weight, mean, and covariance). Additionally, FV yields a fixed-size feature vector, making it a convenient input to any classifier. We explain how to derive FV in our proposed method. Let $X_i = \{\boldsymbol{p}_t \in \mathbb{R}^3,\ t = 1, ..., T\}$ be the set of 3D points of a cluster i, where T denotes the number of points in a cluster. Let $\lambda = (\mu, \Sigma)$ denote Gaussian parameters, where $\mu = (\mu_x, \mu_y, \mu_z)$ and Σ are expected value and covariance matrix of the Gaussian. The proposed method uses one Gaussian to compose GMM, and μ_x and μ_z can be obtained by averaging the x and z coordinates of the points in the pillar respectively, and we use a constant value (1000) for μ_y. For fast computation, we use a unit matrix as Σ. The likelihood of a single 3D point $\boldsymbol{p}$ with the Gaussian density is:

$$u(\boldsymbol{p}) = \frac{1}{(2\pi)^{3/2}|\Sigma|^{1/2}} exp\{-\frac{1}{2}(\boldsymbol{p}-\mu)^T\Sigma^{-1}(\boldsymbol{p}-\mu)\} \quad (1)$$

The Fisher Vector $\mathscr{G}$ is the sum of normalized gradient statistics, computed here for each point $\boldsymbol{p_t}$:

$$\mathscr{G} = \sum_{t=1}^{T} L\nabla log\ u(\boldsymbol{p_t}) \quad (2)$$

The normalized gradient can be written in terms of the variable:

$$\mathscr{G}_\alpha = \sum_{t=1}^{T}(u(\boldsymbol{p}_t) - 1) \quad (3)$$

$$\mathscr{G}_\mu = \sum_{t=1}^{T} u(\boldsymbol{p}_t)\left(\frac{\boldsymbol{p}_t - \mu}{\sigma}\right) \quad (4)$$

$$\mathscr{G}_\sigma = \sum_{t=1}^{T} u(\boldsymbol{p}_t)\left(\frac{(\boldsymbol{p}_t - \mu)^2}{\sigma^2} - 1\right) \quad (5)$$

To construct a feature vector to be fed to the object inference model (explained in Section IV-C), we use the maximum value in the summation of each equation (3)-(5) (7-dimension), the minimum value in the summation of each equation (4)-(5) (6-dimension), and the height (1-dimension) and the number T (1-dimension) of points in each cluster, as well as the obtained

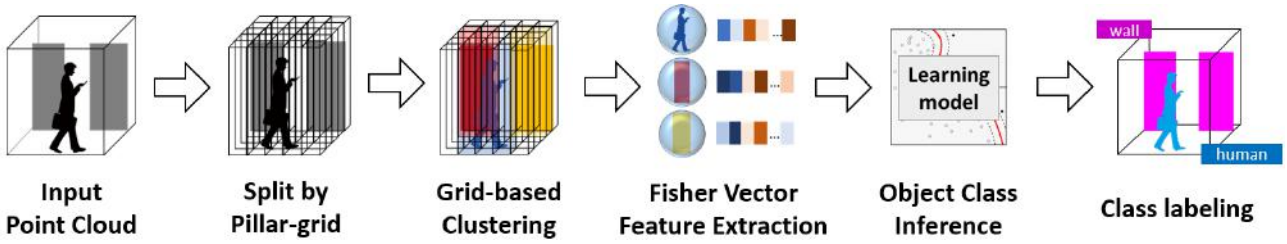

Fig. 3. Object Recognition Procedure

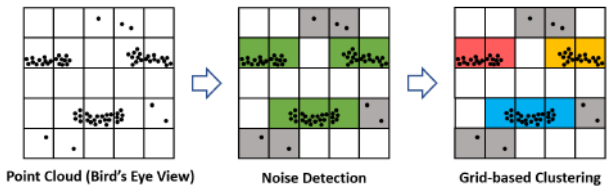

Fig. 4. Proposed clustering (Point cloud viewed from above)

Fisher Vector representation (7-dimension). Consequently, we obtain 22-dimensional feature vector for each cluster as:

$$\mathscr{X}_{input} = \begin{bmatrix} \sum_{t=1}^{T} L\nabla log\ u(\boldsymbol{p_t})|_{\lambda=\alpha,\mu,\sigma} \\ max(L\nabla log\ u(\boldsymbol{p_t}))|_{\lambda=\alpha,\mu,\sigma} \\ min(L\nabla log\ u(\boldsymbol{p_t}))|_{\lambda=\mu,\sigma} \\ max(y|(x_t, y_t, z_t) \in X_i) \\ T \end{bmatrix} \quad (6)$$

We refer to [22] for the details of the formulas and implementation.

C. Object Class Inference

For each cluster (as in Section IV-A), a feature vector in the form of Eq. (6) is obtained. Then, this feature vector is used as an input to a Support Vector Machine (SVM) classification model that identifies the class of each cluster. SVM is a kernel-based machine learning classification model that determines a function of decision boundaries to maximize the margin between existing decision boundaries in the feature space.

The output of SVM is the class (human, wall, furniture, etc.) of the object. We consider the true class label of each cluster as the most frequent label among the points in a cluster. We train the classification model using feature vectors and their corresponding true class labels for each cluster obtained from the annotated point cloud dataset.

V. Evaluation

A. Dataset

A dataset was created to evaluate the system's performance described in Section IV. The *Hitonavi-μ* device was fixed at the height of $1m$ on a tripod, and we acquired point clouds for 13 different scenarios in a $15m \times 25m$ room. Currently, the device is intended for use in indoor environments. It is our ongoing work to make the algorithm work in mobile scenarios. The average frame rate was 10 frames/second. The captured point cloud consists of 12,105 frames, and we created 3D bounding boxes for the objects in all frames for annotation. This annotation process is essential for training and testing the proposed method. The class labels of the objects are "human," "wall," and "chair," and points that do not belong to any of the bounding boxes are considered as noise and are not assigned a class label. The dataset details are shown in Table I.

B. Comparison Method

To highlight the balance between the accuracy and processing speed, we have also implemented a method using a conventional clustering method, DBSCAN, instead of pillar-based discretization and clustering. This method is denoted as *DBSCAN-FV*. More specifically, *DBSCAN-FV* performs feature extraction based on the FV representation for each DBSCAN cluster. This means that the method incurs the processing cost of point-based segmentation but may have better classification accuracy as it captures the features of a whole target object, while ours, denoted as *Pillar-grid FV*, focuses on the part of the features separated by pillars.

C. Experimental Result

The first 80% frames of each scenario were used for training the SVM models of both approaches, and the second 20% frames were used for performance evaluation.

We measured the accuracy and execution time. Average Precision (AP) for each class and mean Average Precision (mAP), which is the average AP for each class, are used for the evaluation.

Execution time refers to the time required for the class inference. The measurement starts after loading all 2400 frames of 3D point cloud data as Python variables and ends when all point clouds have been segmented, and classes have been inferred. We run this process 20 times and calculate the average execution time. The entire process is executed on a Raspberry Pi 4 Model B of *Hitonavi-μ*.

The average precision and the execution time are shown in Tables II and III, respectively. Regarding recognition accuracy, our proposed method, Pillar-grid-FV, achieves almost the same accuracy as DBSCAN-FV. We look into the details later in this section. As for the execution time, we achieve significant improvement. The overall processing time is reduced by 81.3%, and the average fps reaches 59.3. To see the details, we show the breakdown of the processing time of both methods in Fig. 5. As seen, the time required for discretization and grid-based clustering is less than one-ninth of the time required

TABLE I
OUR DATASET DETAILS ∗: MEASURED FROM MULTIPLE ANGLES (30, 60, AND 90 DEGREES)(0 DEGREES WHEN THE LIDAR DETECTION DIRECTION AND THE WALL ARE DIRECTLY OPPOSITE TO EACH OTHER.) ◇: MEASURED FROM MULTIPLE OBJECT DISTANCES ($60cm$, $120cm$, $180cm$)

Scenario		#Frames	Existing objects (M = Move, S = Stationary)								
			Human			Wall			Chair		
Total frames		12105	M	S	(note)	M	S	(note)	M	S	(note)
01	A wall	2272					1	∗			
02	A human	1893		1	◇						
03	A human passing by a wall	397	1		pass		1				
04	A human passing	436	1		pass						
05	A human approaching from a wall	352	1		approach		1				
06	A human approaching	362	1		approach						
07	A human departing to a wall	370	1		depart		1				
08	A human departing	234	1		depart						
09	A chair (front)	2392								1	front◇
10	A chair (side)	1321								1	side◇
11	A wall and a chair	973					1			1	
12	A human and a wall and a chair	572		1			1			1	
13	A human and a wall	531	1				1	45°			

TABLE II
AVERAGE PRECISION

	Human	Wall	Chair	mAP
DBSCAN-FV	96.0	96.1	100	96.2
Pillar-grid-FV (ours)	97.3	91.0	100	96.1

TABLE III
PROCESSING TIME (2400 FRAMES)

	Processing time (s)	frames/second
DBSCAN-FV	216.6	11.1
Pillar-grid-FV (ours)	**40.5**	**59.3**

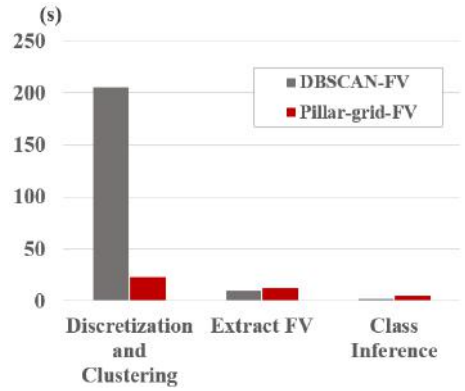

Fig. 5. Comparison of processing time for each module (2400 frames)

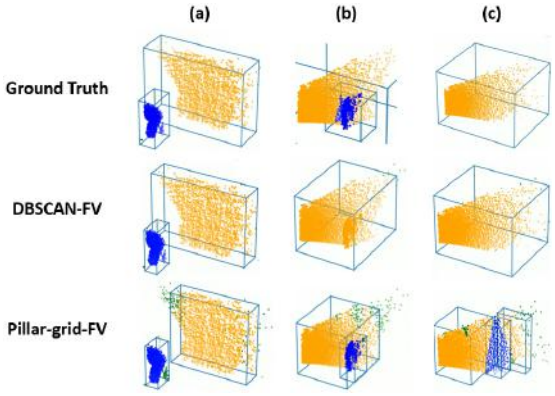

Fig. 6. Ground truths and segmentation results (Points with yellow, blue, green colors refers to wall, human, and noise, respectively).

by DBSCAN clustering, which contributes significantly to the reduction in processing time. On the other hand, the time required for subsequent processing is 1.5x longer for the proposed method, but does not affect a lot to the whole processing speed.

To investigate the results more, the ground truth and the segmentation results are shown in Fig. 6. In frame (a), both point clouds are generally segmented correctly. Frame (b) shows a case where a person is in front of a wall. While the proposed method can separate the wall and the person, DBSCAN-FV cannot do so. This is due to the processing of the clustering algorithm: because DBSCAN is a density-based clustering algorithm, if the point cloud of people and one of the walls are close to each other, they are included in one cluster. On the other hand, the proposed method is not sensitive to the distance between objects. This is because the pillar-grid discretization separates persons and walls appropriately.

Since the clustering method merges adjacent grids into a single cluster, a wall that spans over diagonal grids may often be regarded as multiple clusters. Due to this feature, the Average Precision of the wall is a bit lower than the others. Nevertheless, we achieved 96.1 mAP, which is sufficient for scene recognition.

The proposed discretization and clustering were five times faster than DBSCAN clustering. As described in Section IV-A, the time complexity of the pillar-grid discretization and grid-based clustering is $O(n)$, which contributes significantly to the overall processing time reduction. On the other hand, the proposed method takes longer time for feature extraction and class inference than DBSCAN-FV. This is due to an error in the clustering process of the proposed method. Most of these errors were caused by dividing the data into more clusters than the number of clusters output by DBSCAN. However, the difference in execution time between feature extraction and class inference is $2.5(ms)$ per frame, which has almost no impact on real-time performance, a requirement of the proposed method.

D. Wearable Device Scenario

Hitonavi-μ is intended to be used as a wearable device in some scenarios. While walking, the device is unstable, and the

LiDAR direction should be calibrated.

We acquired the data in the wearable device scenario in the following procedure. One device wearer and one subject stand at a distance of $2.5m$. The wearer wears the device around his neck and starts sensing while standing still. The wearer starts walking in the direction of the subject, passes by the subject's side, and then terminates the acquisition.

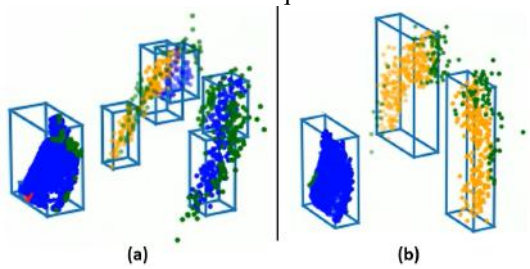

Fig. 7. Segmentation results of acquired data during walking

The segmentation results of the acquired data are shown in Fig. 7. The point cloud (a) was acquired when the device was tilted upward by 20 degrees. The raw point cloud cannot be correctly segmented by the proposed method. On the other hand, point cloud (b) is after the pose calibration, the x-z plane of the point cloud is parallel to the real world ground, and correct object recognition is performed. Due to space limitations, we omit the details of the calibration.

VI. Conclusion

This paper introduced our portable 3D spatial sensing device called *Hitonavi-μ* based on which we proposed a novel computationally-efficient object recognition method. The proposed method utilizes space discretization by pillars and feature extraction and representation based on Fisher Vector, which leads to a low processing cost that enables running on edge devices with limited computational power. We have evaluated our method in terms of recognition accuracy and execution time using our dataset captured by *Hitonavi-μ*. The experimental results showed that the proposed methods achieved 96.1 mAP with a high processing speed as 59.3 frames/second on edge devices. We can conclude that we could achieve a good balance between processing time and recognition accuracy.

We are working on further refinement and tuning of all the modules for facilitating support to visually impaired in in-situ environments.

Acknowledgement

This work was supported by JST, A-STEP Grant Number JPMJTR20RV, Japan.

References

[1] Ministry of Health, Labour and Welfare, Japan. covid-19 contact-confirming application (cocoa). [Online]. Available: https://www.mhlw.go.jp/stf/seisakunitsuite/bunya/cocoa_00007.html

[2] H. Rizk, A. Saeed, and H. Yamaguchi, "Vaccinated, what next? an efficient contact and social distance tracing based on heterogeneous telco data," *IEEE Sensors Journal*, vol. 22, no. 18, pp. 17 950–17 962, 2022.

[3] H. Rizk, T. Amano, H. Yamaguchi, and M. Youssef, "Smartwatch-based face-touch prediction using deep representational learning," in *Mobile and Ubiquitous Systems: Computing, Networking and Services*, T. Hara and H. Yamaguchi, Eds. Cham: Springer, 2022, pp. 493–499.

[4] ——, "Cross-subject activity detection for covid-19 infection avoidance based on automatically annotated imu data," *IEEE Sensors Journal*, vol. 22, no. 13, pp. 13 125–13 135, 2022.

[5] P. Voigtlaender, M. Krause, A. Osep, J. Luiten, B. Sekar, A. Geiger, and B. Leibe, "Mots: Multi-object tracking and segmentation," in *CVPR*, 2019, pp. 7942–7951.

[6] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[7] H. Yamaguchi, A. Hiromori, and T. Higashino, "A human tracking and sensing platform for enabling smart city applications," in *ICDCN Workshops*. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3170521.3170534

[8] X. Zhu, H. Zhou, T. Wang, F. Hong, Y. Ma, W. Li, H. Li, and D. Lin, "Cylindrical and asymmetrical 3d convolution networks for lidar segmentation," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 9939–9948.

[9] H. Rizk, H. Yamaguchi, M. Youssef, and T. Higashino, "Laser range scanners for enabling zero-overhead wifi-based indoor localization system," *ACM Trans. Spatial Algorithms Syst.*, may 2022. [Online]. Available: https://doi.org/10.1145/3539659

[10] ——, "Gain without pain: Enabling fingerprinting-based indoor localization using tracking scanners," in *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '20. New York, NY, USA: ACM, 2020, p. 550–559.

[11] S. Yamada, H. Rizk, and H. Yamaguchi, "An accurate point cloud-based human identification using micro-size lidar," in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 2022, pp. 569–574.

[12] Y. Okochi, H. Rizk, and H. Yamaguchi, "On-the-fly spatio-temporal human segmentation of 3d point cloud data by micro-size lidar," in *18th International IE*, 2022, pp. 1–4.

[13] H. Katayama, T. Mizomoto, H. Rizk, and H. Yamaguchi, "You work we care: Sitting posture assessment based on point cloud data," in *IEEE PerCom Workshops*, 2022, pp. 121–123.

[14] V. Erdélyi, H. Rizk, H. Yamaguchi, and T. Higashino, "Learn to see: A microwave-based object recognition system using learning techniques," in *ICDCN Workshops*. New York, NY, USA: Association for Computing Machinery, 2021, p. 145–150. [Online]. Available: https://doi.org/10.1145/3427477.3429459

[15] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *arXiv preprint arXiv:1612.00593*, 2016.

[16] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *arXiv preprint arXiv:1706.02413*, 2017.

[17] C. R. Qi, O. Litany, K. He, and L. J. Guibas, "Deep hough voting for 3d object detection in point clouds," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[18] S. Kim, S. Kim, J. Lee, and H.-J. Yoo, "A low-power graph convolutional network processor with sparse grouping for 3d point cloud semantic segmentation in mobile devices," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 69, no. 4, pp. 1507–1518, 2022.

[19] H. Liu, R. Liu, K. Yang, J. Zhang, K. Peng, and R. Stiefelhagen, "Hida: Towards holistic indoor understanding for the visually impaired via semantic instance segmentation with a wearable solid-state lidar sensor," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, October 2021, pp. 1780–1790.

[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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[21] S.-S. Li, "An improved dbscan algorithm based on the neighbor similarity and fast nearest neighbor query," *IEEE Access*, vol. 8, pp. 47 468–47 476, 2020.

[22] Y. Ben-Shabat, M. Lindenbaum, and A. Fischer, "3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3145–3152, 2018.