This is a preprint. The DOI for the final published version is available at https://doi.org/10.23919/ICCAS63016.2024.10773172.

Cloud Robotics: Offload scale optimization aided SLAM using reinforcement learning*

Jaeguk Byeon[1], Hyunrok Cha[1], Myeonghwan Hwang[1], Seungha Yoon[1], and Eugene Kim[2]

***Abstract*— There is no doubt that there has been a great development of autonomous driving technology due to an improvement in processing devices regarding vehicular controlling units. However, not only the improvement of the performance of the vehicular controlling units but also the requirements for the absolute amount of the computation cost raised alongside the rapid development of artificial intelligence or core functionalities regarding autonomous driving. As for the countermeasure to dismiss the limited computational issues, cloud-based offloading is considered a potential candidate for a feasible solution. This study introduces a novel cloud offloading approach that uses scale optimization in master-slave architecture based on a real-time simultaneous localization and mapping case scenario. The result shows that the performance of the — increased, and improved in terms of computational cost. It is expected that the suggested framework therefore can be aided in the cost reduction of implementing multiple agents and vehicular applications, especially using a cloud-based approach.**

I. INTRODUCTION

The safety of autonomous driving is the most focused topic regarding social acceptability [1]. In order to achieve such essential social acceptability, a high demand for a safety integrity level is required [2], [3]. However, such a high safety integrity level requires sufficient precision and accuracy against recognition of the environment such as using AI [4]. However, as the number of tasks performed inside an existing vehicle increases, the individual/development unit cost of the vehicle rises, and this is pointed out as a factor that hinders/makes expansion and commercialization difficult.

As for the countermeasure, cloud robotics has been proposed as a way to solve the problem of limited on-board computation in mobile robot systems, and it broadly refers to the process of utilizing cloud-based computing resources [5], [6], [7]. The basic concept of cloud-based offloading for heavy computation tasks can be widely found in the realm of robotic and computer vision [8]. For example, robots can offload processing of sensor inputs such as video, audio, or LIDAR, and specifically, the offloading approach has been used for simultaneous mapping and localization, recognition, and speech processing [9], [10], [11], [12].

However, as Chinchali et.al. mention, a determination of network offloading policies heavily relies on cloud resources and unforeseen network quality degradation can result in overall performance degradation due to the latency [13]. What should be mentioned is that not all can be offloaded to the cloud region depending on the type of sensor properties. Especially when it comes to dealing with relatively large amounts of data samples such as high-definition video data (HD), or LiDAR (point cloud samples) from the viewpoint of wireless communication, utilizing cloud offloading may cause significant performance drops.

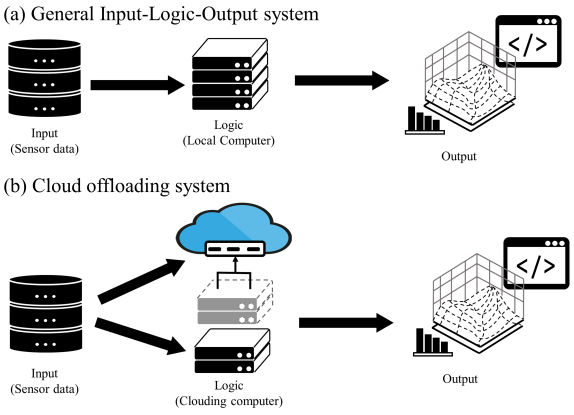

Fig. 1. Comparison of general Input-Logic-Output system and a cloud offloading system

A. Contributions

In this study, we focus on LiDAR offloading and its treatments from the viewpoint of mapping and localization. Our core idea is to rescale the sample size of the interest so as not to hamper the performance of the overall system by optimizing it via reinforcement learning. In order to organize and optimize the reinforcement learning-aided neural networks, datasets were established to give a reward when there was no degradation in sample receiving intervals. As a result, our proposed model showed that it can avoid communication loss or degradation of mapping performance by re-scaling the point cloud samples dynamically.

II. METHODOLOGY

Figure 1 shows a schematic diagram of the basic concept comparing a general Input-Logic-Output system and the cloud offloading system. It can be noticed that thanks to the offloading of computation tasks at the logic stage, the performance of local computation performance may lightened while maintaining the overall performance of the system.

*This work was supported by "Development of Core Technologies for a Working Partner Robot in the Manufacturing Field" (KITECH EO-24-0007)

[1]Authors are with the Purpose-Based Mobility group, Seonam Division, Korea Institute of Industrial Technology, Gwangju 61012, South Korea wornr7390@kitech.re.kr

[2]Eugene Kim is the corresponding author with the Purpose-Based Mobility group, Seonam Division, Korea Institute of Industrial Technology, Gwangju 61012, South Korea egkim@kitech.re.kr

This is a preprint. The DOI for the final published version is available at https://doi.org/10.23919/ICCAS63016.2024.10773172.

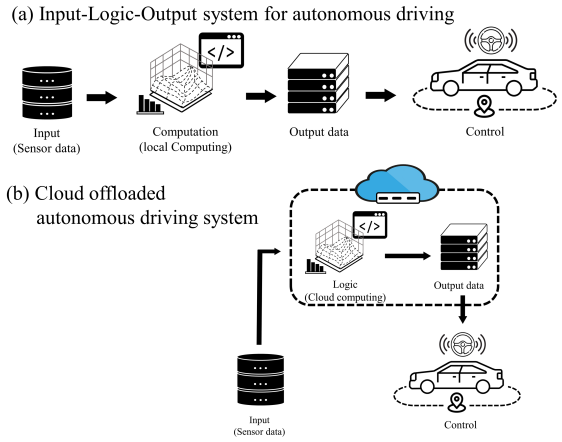

Fig. 2. Comparison of the local Input-Logic-Output system for autonomous driving system, and the cloud offloaded autonomous driving system

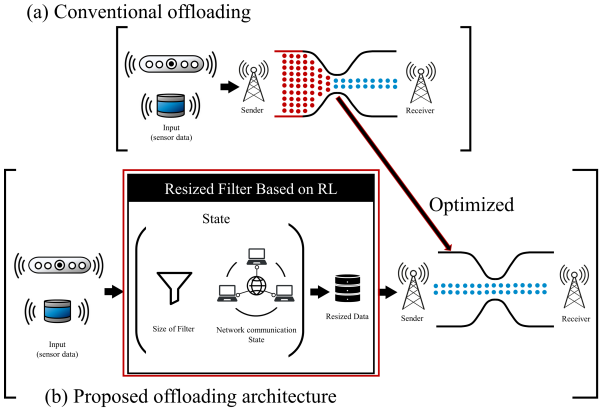

Fig. 3. Concept comparison of the typical offloading method and that of the proposed offloading architecture. In order to dynamically optimize the amount of the received samples of the sensor data, a resized filter was proposed based on reinforcement learning

Fig. 2 represents a concept comparison of a local Input-Logic-Output system and the cloud-offloaded autonomous driving system. It can be noticed that on behalf of the computation task allocated to the local computation unit, cloud computation was utilized to make the autonomous driving system lighter.

Fig. 3 shows the problem of typical offloading and the merit of the proposed offloading method using a resized filter when transmitting large-scale sensor samples. What can be known is that the typical offloading method creates a bottleneck when transmitting large-scale data samples when it gets higher than the limit of network bandwidths. Therefore, we aim to design a novel resize filter approach that can optimize the scale of sensor samples dynamically according to the network bandwidth that can aid in solving the bottleneck problem.

A. Voxel grid filter

In this study, the Voxel grid filter was utilized to resize the scale of the sensor samples [14]. As Fig. 4 shows, the

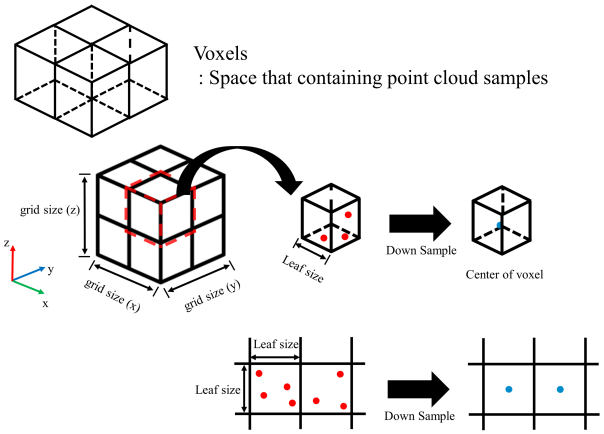

Fig. 4. Illustration of voxel grid filter used for the downsampling for the point cloud samples

voxel grid approach offers several advantages, particularly in the context of 3D sample processing and LiDAR point cloud downsampling. The core idea of the voxel grid approach is to divide the space into each voxel which has the same leaf size length to contain corresponding point cloud samples.

Let $P = \{\mathbf{p}_i \mid \mathbf{p}_i \in \mathbb{R}^3, i = 1, 2, \ldots, N\}$ be the original set of LiDAR points, where $\mathbf{p}_i = (x_i, y_i, z_i)$ represents the coordinates of the i-th point measured by the LiDAR sensor. Then, the voxel grid is divided with each voxel size v. Each point $\mathbf{p}_i$ is mapped to a voxel V_{ijk} based on its coordinates:

$$V_{ijk} = \left\{ \mathbf{p}_i \mid \left\lfloor \frac{x_i}{v} \right\rfloor = i, \left\lfloor \frac{y_i}{v} \right\rfloor = j, \left\lfloor \frac{z_i}{v} \right\rfloor = k \right\}. \quad (1)$$

Next, for each voxel V_{ijk} containing M points, the centroid $\mathbf{c}_{ijk}$ is computed as follows:

$$\mathbf{c}_{ijk} = \frac{1}{M} \sum_{\mathbf{p}_i \in V_{ijk}} \mathbf{p}_i, \quad (2)$$

where $\mathbf{c}_{ijk} = (x_{ijk}, y_{ijk}, z_{ijk})$.

Finally, the downsampled point cloud P' represents the set of centroids $\mathbf{c}_{ijk}$ for all non-empty voxels:

$$P' = \{\mathbf{c}_{ijk} \mid V_{ijk} \neq \emptyset\}. \quad (3)$$

B. Sensor sample resize via reinforcement learning

Reinforcement learning was used to choose the optimized scale of sample, which is adapted to various network bandwidths.

We express the sample offloading problem as a MDP (Markov Decision Process) :

$$M^{\text{offload}} = (s^{\text{offload}}, a^{\text{offload}}, r^{\text{offload}}, n), \quad (4)$$

where s^{offload} is the state space, a^{offload} is the action space, $r^{\text{offload}} : s^{\text{offload}} \times a^{\text{offload}}$ is a reward function, and n is sequence of sample.

State sapce: This space contains information indicating the communication status and the previous size of the voxel to choose appropriate size of voxel at *Action*. We choose

the receive time as representing the communication status. The $state$ in the offloading MDP was defined

$$s_n^{\text{offload}} = [T_{n-1}, v_{n-1}], \tag{5}$$

where T is time that the master pc received a sample.

$Action\ space$: This space includes the action of decreasing, increasing, or maintaining the size of voxel, based on the $state$. Increasing the size of voxel means reduce the number of point clouds. This is the same thing as lowering the sample size. On the contrary, decreasing the size of voxel means maintaining a larger number of point clouds. The $action$ in the offloading MDP was defined

$$a_n^{\text{offload}}(s_n) = \begin{cases} 0, & v_{n-1} - 0.02 \\ 1, & v_{n-1} + 0 \\ 2, & v_{n-1} + 0.02. \end{cases} \tag{6}$$

$Reward\ function$: The $Reward\ function$ was classified according to the communication status. The criterion is the T, and the reference T is 0.11 s. The reason is that its frequency of the sample topic is 9.7 $\sim$ 9.8 Hz when the master PC receive the sample topic without delay. Substituting that frequency with time is less than about 0.11 s. So, the reference T was chosen as 0.11 s. The original size of sample(S) and resized size of sample(S') were used to give higher reward when maintain close to the original sample size.

When $T < 0.11$s, reward function(r^{offload}) was defined

$$r_n^{\text{offload}}(s_n, a_n) = \begin{cases} a_n = 0, & \frac{S'_n}{S_n} \\ a_n = 1, & 0 \\ a_n = 2, & -\frac{S'_n}{S_n}, \end{cases} \tag{7}$$

and when $T \geq 0.11$s, reward function(r^{offload}) was defined

$$r_n^{\text{offload}}(s_n, a_n) = \begin{cases} a_n = 0, & -\frac{S'_n}{S_n} \\ a_n = 1, & (v_n - v_{max})\frac{S'_n}{S_n} \times 10 \\ a_n = 2, & 0, \end{cases} \tag{8}$$

where v_{max} is the maximum size of voxel.

III. Experiment

A. General setup

In order to use cloud computing, it is necessary to connect the data obtained from the sensor to the computing unit, and this can generally be accomplished through ROS (Robot Operating System) [15]. The experimental setup utilized the ROS Noetic framework running on the Ubuntu 20.04 operating system. It consisted of two master-slave PC configurations to facilitate distributed computing. Also, LiDAR equipment (Velodyne, VLP-16, San Jose, California, United States) was used to measure the 3D point cloud samples. The experimental procedures were conducted within a WiFi network environment. The IP Time Router, designated as Model N704, played a pivotal role in facilitating data transmission within the WiFi network (Max 2,200 Mb/s).

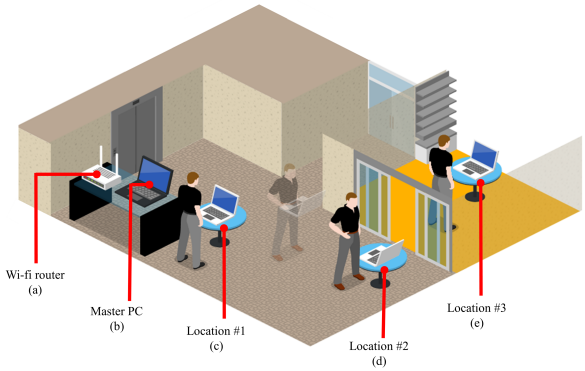

Fig. 5. Experimental condition for gathering dataset for training the proposed algorithm. The Wi-fi router (a) and Master PC (b) were placed fixed. Slave PC was moved along the location fro #1$\sim$#3 (c),(d),(e)

B. Experiment conditions

Overall, Fig. 5 shows the illustration of gathering the dataset for training the proposed algorithm.

1) Comparison of bandwidth according to the number of LiDAR scans with limited bandwidth: The default settings for measuring Velodyne LiDAR require about 4.3 MB/s of bandwidth. As for the comparison, double and triple times of LiDAR samples were transmitted to the receiver. It is assumed that according to the number of transmitted LiDAR scan samples, each allocated bandwidth is reduced.

2) Comparison of received frequency according to the number of LiDAR scans with limited bandwidth: As for the analysis, double and triple times of LiDAR samples were transmitted to the receiver. It is assumed that according to the number of transmitted LiDAR scan samples, each frequency of the received samples is reduced.

3) Comparison of received time according to the number of LiDAR scans with and without the proposed method: As for the comparison, received time analysis according to the network quality was conducted. More precisely, mapping performance[1] with and without the proposed model was tested by moving along the location #1 ($\sim$-40 dBm), location #2 (-62$\sim$-48 dBm), and location #3 ($\sim$-70 dBm). It is assumed that the proposed model is less affected by the quality of the network.

Among the reinforcement learning methods, DQN was used, and the learning rate was 0.0005, the optimize was Adam, the discount factor was 0.98, the batch size was 64, and the voxel size range was [0.01 $\sim$ 0.07]. The voxel size 0.01 maintains about 90% of the raw sample size, and 0.07 maintains about 40% of the raw sample size. And determining the state, action, and reward for one sample was designated as one episode. In response, 5,000 samples were received for each location, and a total of 300,000 episodes were conducted. As the neural network structure of DQN,

[1]Lego-Loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain [16]

This is a preprint. The DOI for the final published version is available at https://doi.org/10.23919/ICCAS63016.2024.10773172.

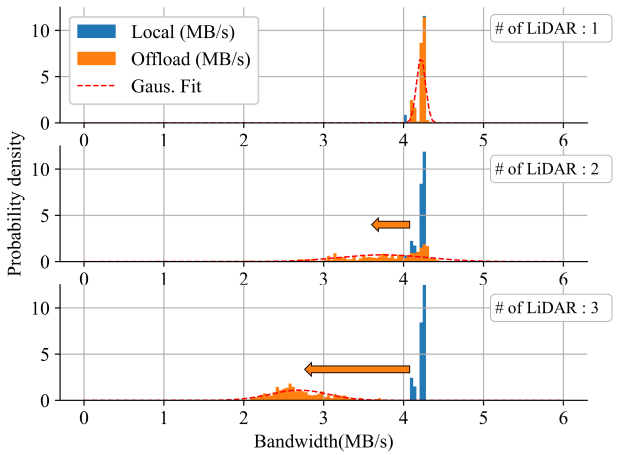

Fig. 6. Bandwidth analysis according to the size of the transmitted LiDAR scan samples. As the number of LiDAR sensors increases, allocated bandwidth decreases (Orange)

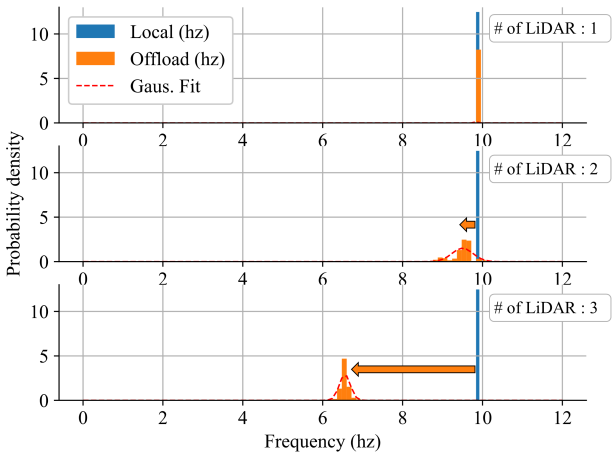

Fig. 7. Frequency analysis according to the size of the transmitted LiDAR scan samples. As the number of LiDAR sensors increases, the frequency of received samples decreases (Orange)

three linear layers were used, two were hidden layers and one was the final output layer. The initial value of the voxel size and the receive time are v_0=0.05 and T_0=0, respectively.

IV. RESULTS AND DISCUSSIONS

A. Bandwidth analysis according to the size of the transmitted LiDAR scan samples

Figure 6 shows the comparison of bandwidth probability density according to the number of LiDAR sensors in the network. The result shows that as the number of LiDAR increases, the mean allocated bandwidth for the offload decreases. Contrary, as the number of LiDAR increases, the standard deviation of the offload allocated bandwidth distribution increases. This indicates that as the amount of transmitted sample size increases, offload performance can be degraded when it overcomes the allowed bandwidth.

B. Frequency of receiving sample analysis according to the size of the transmitted LiDAR scan samples

In Fig. 7, the outcomes of frequency analysis are depicted concerning the incremental addition of LiDAR sensors. It becomes evident that as the number of LiDAR sensors increases, the constrained bandwidth imposes a bottleneck on the network, consequently resulting in performance degradation.

C. Comparison of performance with and without the proposed method

Figure 8 shows performance comparison when using with and without the proposed offload policy while the network quality is dynamically changing. Due to the quality of the network change (High to Low), the performance of the raw cloud offload method degrades. However, by use of the proposed approach, due to the sample size of the point cloud dynamically changing, received time is less affected compared to that of the raw offload policy.

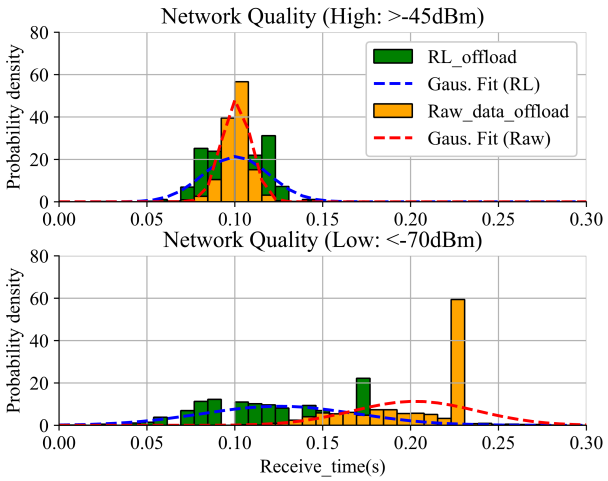

Fig. 8. Comparison of performance with and without the proposed method. The received time was compared when the receiver was laid on decent network quality and poor network quality. The result shows that our proposed method is less affected by the quality of the network

Fig. 9 shows the result of Lego-Loam mapping on the local computer when not utilizing the offloading policy which had 229,213 points registered. On the other hand, Fig. 10 shows the result of the raw offloading policy which had 92,664 points registered (60% decreased). On the contrary, as Fig. 11 shows the result of the proposed offloading policy which had 165,184 points registered (28% decreased). From the result, it is evident that the proposed method can resist the dynamic network quality compared to the raw cloud offload policy. Finally overall comparison of the experiment is briefly summarized in Table I.

V. CONCLUSIONS

Due to the rapid advancement of in-vehicle control units, the era of autonomous driving has emerged. However, main-

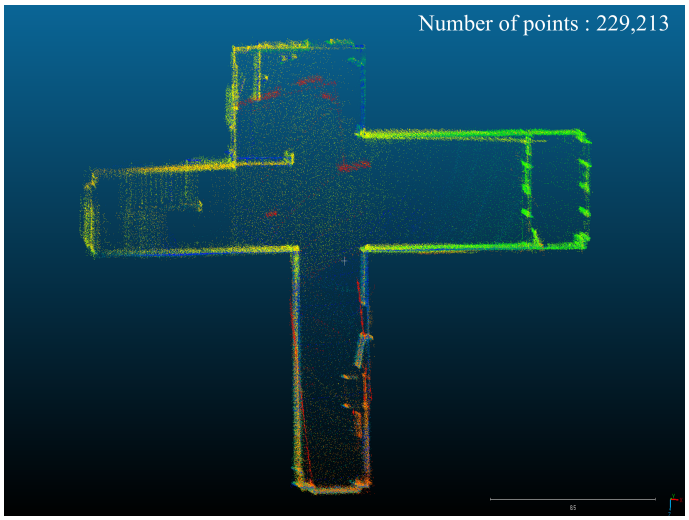

Fig. 9. Result of Lego-Loam mapping on local computer. The total number of points registered to the map was 229,213

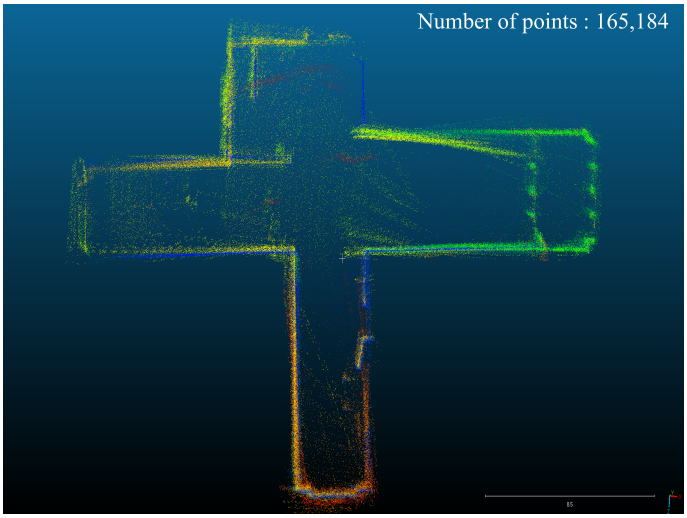

Fig. 11. Result of Lego-Loam mapping using cloud mapping (Proposed). The total number of points registered on the map was 165,184. It depicts that the registered points decreased by 28%

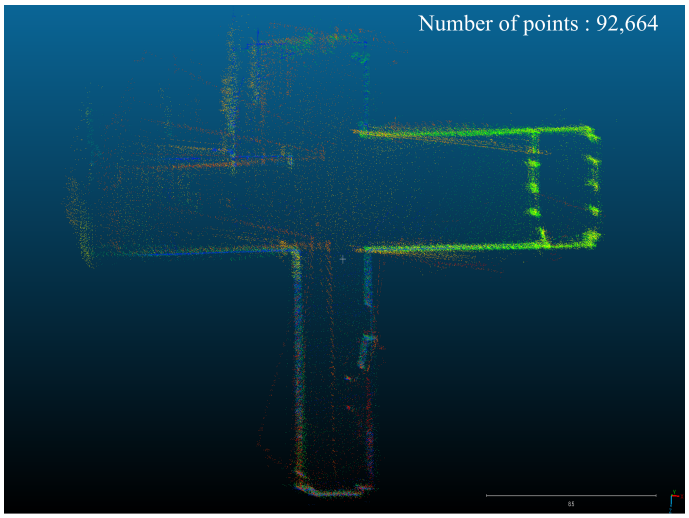

Fig. 10. Result of Lego-Loam mapping using cloud mapping (Raw). The total number of points registered on the map was 92,644. It depicts that the registered points decreased by 60%

TABLE I

SUMMARY OF COMPARISON OF THE WITH AND WITHOUT THE PROPOSED CLOUD OFFLOAD POLICY, WHERE S IS THE SAMPLE SIZE OF THE POINT CLOUD, $N(S)$ REPRESENTS THE NUMBER OF THE POINT CLOUD SAMPLES, AND $\bar{T}$ DEPICTS THE AVERAGE TIME TO RECEIVE THE SAMPLES

	S	$N(S)$	$\bar{T}$
Local	386.48	954	0.10
RL (no delay)	302.84	954	0.10
Raw (no delay)	386.48	954	0.10
RL (delay)	206.46	823	0.118
Raw (delay)	213.64	528	0.184

taining accuracy to ensure safety is deemed imperative. Cloud computing, as a means to compensate for insufficient computing power, holds vast potential. It not only supplements inadequate computing power but also holds promise for successful utilization in low-cost robots and similar devices.

This paper experiments with dynamic resizing for cloud offloading of sensor data with large capacity, such as LiDAR, to fully leverage wireless network environments within the limits of network bandwidth. The proposed method employs reinforcement learning, a subfield of machine learning, to find optimal resizing parameters. Results demonstrate that this approach yields faster and higher-quality SLAM results compared to conventional cloud offloading methods.

Our research extends beyond vehicular applications, confirming the effectiveness of the proposed dynamic scaling method for large-capacity sensor data, such as LiDAR, in cloud environments. As future work, research is needed to detect and secure reliable communication in fluctuating network quality, not only for LiDAR but also for HD cameras and similar devices.

REFERENCES

[1] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.
[2] Road vehicles – Functional safety, ISO 26262:2018.
[3] Functional Safety of Electrical-Electronic/Programmable Electronic Safety-Related Systems, IEC 61508:2010.
[4] SAE-J2735, "V2x communications message set dictionary," 2020.
[5] J. Ariza, M. Jimeno, R. Villanueva-Polanco, and J. Capacho, "Provisioning computational resources for cloud-based e-learning platforms using deep learning techniques," *IEEE Access*, vol. 9, pp. 89 798–89 811, 2021.
[6] K. Goldberg and B. Kehoe, "Cloud robotics and automation: A survey of related work," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2013-5*, pp. 13–5, 2013.
[7] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Transactions on automation science and engineering*, vol. 12, no. 2, pp. 398–409, 2015.

[8] G. Hu, W. P. Tay, and Y. Wen, "Cloud robotics: architecture, challenges and applications," *IEEE network*, vol. 26, no. 3, pp. 21–28, 2012.

[9] P. Zhang, H. Wang, B. Ding, and S. Shang, "Cloud-based framework for scalable and real-time multi-robot slam," in *2018 IEEE International Conference on Web Services (ICWS)*. IEEE, 2018, pp. 147–154.

[10] B. A. Erol, S. Vaishnav, J. D. Labrado, P. Benavidez, and M. Jamshidi, "Cloud-based control and vslam through cooperative mapping and localization," in *2016 World Automation Congress (WAC)*. IEEE, 2016, pp. 1–6.

[11] C. Sennersten, A. Morshed, M. Lochner, and C. Lindley, "Towards a cloud-based architecture for 3d object comprehension in cognitive robotics," in *The 6th International Conference on Advanced Cognitive Technologies and Applications*, 2014.

[12] K. Sugiura, Y. Shiga, H. Kawai, T. Misu, and C. Hori, "A cloud robotics approach towards dialogue-oriented robot speech," *Advanced Robotics*, vol. 29, no. 7, pp. 449–456, 2015.

[13] S. Chinchali, A. Sharma, J. Harrison, A. Elhafsi, D. Kang, E. Pergament, E. Cidon, S. Katti, and M. Pavone, "Network offloading policies for cloud robotics: a learning-based approach," *Autonomous Robots*, vol. 45, no. 7, pp. 997–1012, 2021.

[14] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1–4.

[15] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[16] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.