Practical Collaborative Perception: Design, Implementation, and Evaluation

Jinjiang Wang*, Guofeng Ren*, Fangbing Zhou†, Hui Zhang‡ , Guiyang Luo*, Quan Yuan*, Jinglin Li*
*Beijing University of Posts and Telecommunications, Beijing, China
†South China Normal University, Guangzhou, China
‡Beijing Jiaotong University, Beijing, China
Email: *{wangjinjiang, renguofeng, luoguiyang, yuanquan, jlli}@bupt.edu.cn,
†fangbing5178@163.com, ‡huizhang1@bjtu.edu.cn

***Abstract*—Multi-agent collaborative perception (CP) has recently garnered significant attention due to its potential in extending perception range and enhancing perception robustness. However, most existing CP methods have only been validated in simulations, and as far as we know, there is currently no practical CP system available. To address this gap, this paper focuses on the design, implementation, and evaluation of CP in real-world scenarios. In order to conserve communication resources, our approach advocates sharing partial point clouds rather than complete data. Furthermore, to improve computational efficiency, we propose a voxelization strategy to compute the blind zone and the required point clouds. Experiments conducted on several small autonomous vehicles demonstrate the efficiency of communication and computation.**

***Index Terms*—collaborative perception, autonomous driving, robot platform**

I. Introduction

As one of the key functionality of autonomous driving, perception often faces challenges in single vehicle scenarios. With the introduction and widespread application of technologies like 5G and V2X (vehicle to everything), researchers have proposed collaborative perception techniques. Collaborative perception enables real-time sharing of sensor data between vehicles and roadside devices to fulfill the information requirements of decision algorithms for long-range and blind-spot situations, thereby enhancing its robustness.

The types of data shared in cooperative perception are usually divided into raw sensor data (early cooperation) [1]–[3], intermediate features of neural networks (intermediate cooperation) [4], [5] and detection results (late cooperation) [6]. However, the methods of transmitting intermediate features and detection results are often a compromise with communication bandwidth. Processing raw data always introduces information loss, and the processed data lacks diversity in format. In contrast, raw sensor data can provide the maximum amount of information and a universal data format, widely supporting other algorithms.

Existing early cooperation algorithms are usually tested on simulators or public datasets. This is to some extent understandable due to the high costs and risks associated with real road testing. Nevertheless, the problem with these approaches is that issues such as signal interference and fluctuations between vehicles do not exist or cannot be accurately simulated compared to real road environments. When deploying such algorithms, they usually encounter the "reality gap" [7] problem, meaning that they require additional sim2real [8] algorithms for deployment in real environments, which can be costly and complex.

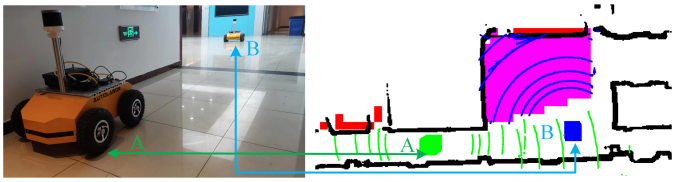

Fig. 1. Two running robotic platforms, and the real-time visualization front-end of vehicle A

In this paper, we introduce practical collaborative perception (PCP), a system that advocate early collaborative perception, which is built upon mobile robot platforms. PCP applies voxelization algorithm to efficiently compute: 1) the blind zone, which is a part of area that requires shared data from neighboring vehicles; 2) The shared points for each neighbor vehicle on the blind zone. Furthermore, we have implement and evaluate PCP with several autonomous vehicles, each equipped with mobile robot platform Autolabor Pro1, RoboSense Helios16P LiDAR, GeForce RTX 4060, Rion AH200C IMU, and wifi interference. The evaluation experiments demonstrate the communication and computational efficiency.

II. Related Work

In early collaboration, sensor data from multiple agents is first aggregated to create a comprehensive view, followed by object detection using this holistic perspective. This approach effectively addresses occlusion and long-range issues that arise in single-agent perception. Moreover, raw sensor data has a basic, fundamental, and universal format that can flexibly support a wide range of CAV applications [9].

Aoki *et al.* [10] introduce a cooperative perception scheme that employs deep reinforcement learning [11] for data transmission selection, mitigating network load in vehicular communication networks and improving communication reliability. Zhang *et al.* [3] create a more comprehensive and higher-resolution view by merging individual vehicle views, transmitting raw sensor data to an edge server. Arnold *et al.* [2] also

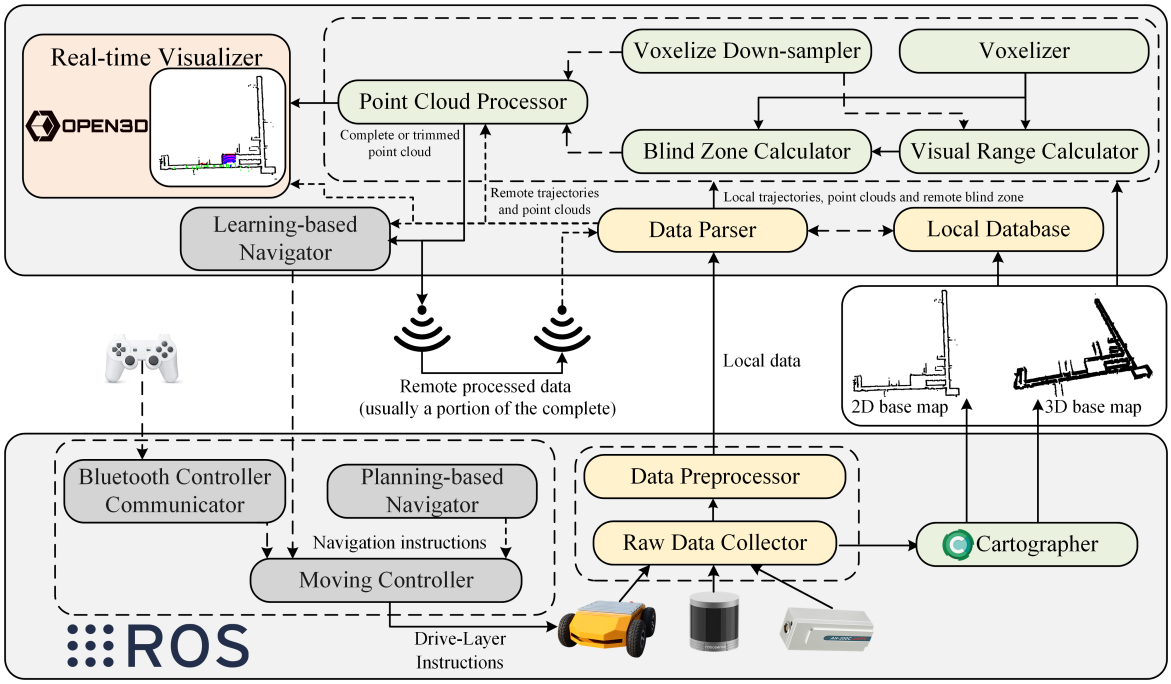

Fig. 2. The Architecture of PCP. The underlying ROS system primarily handles the interactions between the external environment and hardware, and it provides basic SLAM mapping and navigation capabilities through two built-in algorithm libraries, Cartographer and ROS-Navigation. The upper-layer front-end application manages communication and modularly runs complex perception and navigation algorithms. Relevant content will be visualized in real-time within Open3D. The dashed lines represent optional functions, indicating that data flow may not occur.

integrate point clouds from various spatially diverse sensing points before performing detection. Due to the considerable volume of raw sensor data, cooperative communication is essential for higher spectrum efficiency. Yu *et al.* [12] investigate the performance gap between distributed and centralized C-V2X scheduling concerning achievable throughput and communication efficiency in CP. However, they overlook sensor data characteristics and use dummy perception data for simulation purposes.

Additionally, most methods recommend sharing all sensor data among vehicles, which is highly inefficient. For example, in uncertainty-aware localization, only features that contribute to object location inference are necessary, and features of objects with high uncertainties among vehicles can be prioritized for transmission to improve real-time localization accuracy.

To the best of our knowledge, our research paper represents a pioneering effort in enhancing the efficiency of early collaborative perception through the implementation of a voxelization strategy. Furthermore, we have validated our algorithm using real-world data collected in an actual communication environment. Our algorithm showcases a substantial edge in terms of overall communication metrics and computational efficiency.

III. PCP

In this section, we introduce the architecture of PCP, demonstrate its practical performance, and propose a simple and effective algorithm for collaborative perception.

A. Architecture

The overall system architecture of PCP is illustrated in Fig. 2. Its hardware components consist of a mobile robot platform, an inertial measurement unit (IMU), a LiDAR, and a computer (laptop or ITX with a mobile power supply). The mobile robot platform provides adjustable mobility for the entire system, and the built-in odometry system assists with localization. The IMU, a motion sensor based on gyroscopes and accelerometers, serves as the primary localization device for PCP. The LiDAR is currently the sole device in the system for external perception information acquisition.

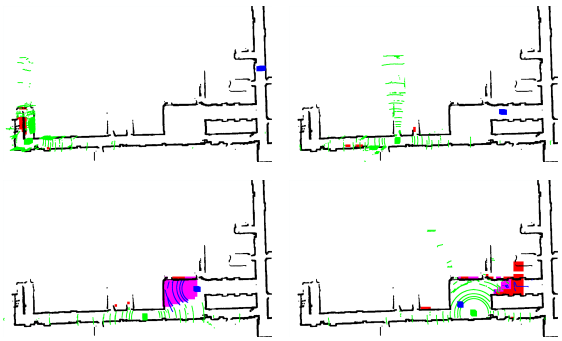

Fig. 3. The front-end of the PCPs running in real-time indoor scenario

In terms of software architecture, PCP utilizes the Robot Operating System (ROS) for data/instructions exchange between the driver layer and the underlying hardware. Due to the modularity of ROS, it allows for the interchangeability

of underlying hardware as long as the hardware provides a ROS driver. During the development process, we have replaced the LiDAR and IMU several times, ensuring the system's high compatibility with different hardware. In the upper layer, to simplify system extension and real-time testing, we have developed a front-end program based on the open-source 3D data processing library Open3D [13]. This program enables most operations on raw data, such as processing, sharing, and visualization, to be script-based using Python. The logic of interacting with the hardware is hidden in the ROS side, and control signals for the hardware can be sent to ROS through pre-packaged APIs in the front-end, significantly reducing the effort required to extend the system's capabilities.

B. SLAM Mapping

Having a robust localization algorithm of cartographer, PCP allows for the data collection and visualization without a map. However, in most cases, a base map makes the real-time visualization of collaborative perception more intuitive, while improving the accuracy of the vehicle's localization. To ensure generality, we allow some algorithms that require map information [14], [15] to run in the system as well. We choose the relatively versatile [16] SLAM algorithm cartographer [17] to build the map. The original output of the cartographer is a 2D occupancy grid map, but for generality, we made modifications to cartographer to support the output of a 3D map by overlaying optimized historical point clouds.

C. Collaborative Perception Algorithm

During the testing runs, we observed that sharing the complete raw point cloud data in real-time posed challenges in terms of parsing speed and bandwidth limitations. This leads to a decrease in the system's operational speed, resulting in information loss. To address this issue, we implement an optional grid-based point cloud sharing algorithm. This algorithm not only improves the sampling efficiency of the system, but also reduces the amount of communication bandwidth required. Additionally, it serves as a foundation for future enhancements in collaborative perception within the system.

We believe that retaining the partial point cloud with the highest information content for real-time decision making is important. However, during operation, at least 80% points of the point cloud sent from vehicle B to vehicle A overlaps with the base map or is within the visual range of vehicle A itself, resulting in a significant amount of communication and computation redundancy. Therefore, we propose an algorithm to ensure the real-time performance of the system.

We design two trimming schemes: 1) simply perform voxel down-sampling on the entire point cloud, which implies a uniform loss of information; 2) transmit only a portion of the point cloud within a specific region, which we believe can provide the maximum additional information. We provide modular implementations for both schemes, allowing for the customization and combination of the two methods and their internal optimization algorithms through parameterization.

1) Voxelization: voxelizer is an optional component in our algorithm. The basic idea of voxelization is to treat the point cloud as a cube and divide it into N grids. Firstly, we manually choose a grid size c. Then, taking the x-axis as an example, on the x-axis with a distribution range of $[x_{min}, x_{max}]$, the number of grids to be divided along the x-axis can be calculated as

$$N_x = (x_{max} - x_{min})/c + 1. \tag{1}$$

After that, we can divide the x-axis into N_x intervals with a spacing of c as one dimension of the grid, and so on, to divide a three-dimensional cube into $N = N_x * N_y * N_z$ grids.

In subsequent algorithms, we have multiple requirements to traverse all points in a point cloud. However, it's worth noting that a single frame of a point cloud typically consists of tens of thousands of points, resulting in a significant computational burden. To optimize this issue, we provide an optional voxelization option in the corresponding algorithm. When enabled, the point-by-point or point-by-grid traversal algorithm becomes a grid-by-grid traversal algorithm. Specifically, for a point cloud with a bounding box size of $X * Y$, the number of traversals is reduced from tens of thousands to

$$n = X * Y/c^2, \tag{2}$$

which is generally much lower than the number of points in the point cloud.

2) Visual Range Calculator: we define the visual range as the distance of the farthest points that the LiDAR can return in each direction. To calculate the blind zone of a LiDAR, we first need to determine its visual range in each real-time frame. The initial step is to voxelize the base map and create a grid-based representation. To enhance computational efficiency, we designate this representation as two-dimensional. Based on this, we assume that the position of the LiDAR is (x_a, y_a), and we randomly select a point (x_b, y_b) from the point cloud. We use the Bresenham line algorithm [18] to calculate the grids the line between (x_a, y_a) and (x_b, y_b) covers and iterate through all points in a frame of the point cloud to obtain the dynamic visual range of the LiDAR for each frame.

The Bresenham line algorithm is used to calculate the pixel points covered by the line segment between two points on the screen to draw a straight line. For the line $y = mx + b$ between given points (x_a, y_a) and (x_b, y_b), every time this line passes through the edge of a grid, that is, for each integer x_i between x_1 and x_2, choose a grid closest to $y = mx_i + b$ (measured by the distance d_1, d_2 between (x_i, y) and the grid points (x_i, y_i) and $(x_i, y_i + c)$, where c is the grid size) as the next drawing object. Repeat the calculate from x_a to x_b. Suppose $m \in (0, 1)$, then we calculate

$$d_1 - d_2 = (y - y_i) - (y_i + \Delta y - y) = 2y - 2y_i - c, \tag{3}$$

if $d_1 - d_2$ is less than 0, we choose the grid where (x_i, y_i) is located. Otherwise, we choose the grid where $(x_i, y_i + c)$ is located. By repeating this process on each point, we can obtain a grid map that represents the visible range of the LiDAR at

this frame. It should be noted that the above formula only applies to the case of $m \in (0, 1)$, and for other cases, the calculation is performed in the same way after operations such as coordinate axis flipping and swapping, etc.

Algorithm 1 Collaborative Perception Algorithm

INIT:
INPUT: base map M
$MapGrids$ = Voxelization(M)
For vehicle A**:**
INPUT: real-time position (x_a, y_a) of A, point cloud P_A from A, $MapGrids$, required range L of A, grid size c in voxelization
for every (x_i, y_i) in P_A **do**
get $CoveredGrids$ from Bresenham$((x_a, y_a), (x_i, y_i))$
end for
summarize all the $CoveredGrids$ in $MapGrids$ as $VisualRange$
get $TraversableGrids$ from closed areas in $MapGrids$
get $RequiredRegion$ from the $L * L$ square region around (x_a, y_a)
$BlindZone$ = $RequiredRegion \cap TraversableGrids - VisualRange$
return $BlindZone$
For vehicle B**:**
INPUT: $BlindZone$ from A, point cloud P_B from B, basemap M, distance threshold $Thres$
Get $KDTree$ of M
for every (x_i, y_i) in P_B **do**
if (x_i, y_i) not in $BlindZone$ **then**
delete (x_i, y_i) from P_B
continue
end if
get $NearestDistance$ to (x_i, y_i) from $KDTree$
if $NearestDistance \leq Thres$ **then**
delete (x_i, y_i) from P_B
end if
end for
return P_B

3) Blind Zone Calculator: taking a network consisting of two vehicles A and B as an example, firstly, we use the voxelized grid base map from the visual range calculator, extract the area it encloses, and perform voxelization again, which represents the traversable area in the base map. Centered on A, extract a square with a side length of L from the traversable area as A's required region. Finally, we eliminate the visible range coverage from A's required region, resulting in a voxelized blind zone grid.

Based on this, A sends the blind zone grid to B, and the real-time point cloud collected by vehicle B only retains points within the blind zone range and sends them to A. In a collaborative perception network consisting of N vehicles, the above calculation operation is performed once for each vehicle, and the communication operation is completed $N-1$ times for each vehicle. Furthermore, the required region size L^2 of any vehicle is individually adjustable, allowing us considerable flexibility in controlling the algorithm.

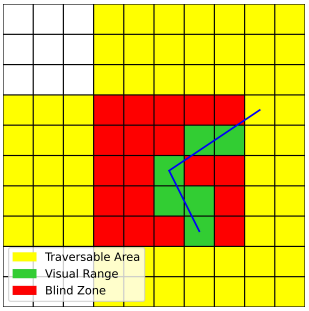

Fig. 4. An example of the algorithm with the assumption L=5. Vehicle A sends the red grid to vehicle B, and vehicle B only returns the points within the red grid

After that, A sends the blind zone grid to B. B traverses each point in the real-time point cloud, determining whether it is within the blind zone, in order to delete all points outside the blind zone. Then, we construct a K-D tree [19] for the base map and query the nearest neighbor of every remaining points with an expected complexity of $O(logn)$, deleting all points with a distance smaller than a threshold from any point in the base map. Finally, we obtain a point cloud within the blind zone of A, with overlapping points with the base map removed. B then sends the processed point cloud to A.

IV. Experiments

A. Test Runs

We conducted systematic testing in both outdoor and indoor scenarios and collected two datasets. The indoor dataset was collected in a designed environment with two sets of systems for collaborative perception. To showcase the capabilities of the system, we utilize the indoor scene dataset for experiments.

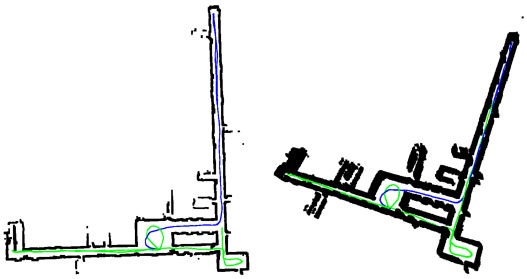

Fig. 5. The 2D and 3D base map with robot trajectory of the indoor scene

The indoor scene dataset was acquired in a typical corridor environment. We ran two sets of vehicles and shared point cloud data in real-time to achieve collaborative perception. The trajectories of the two vehicles are depicted in Fig. 5.

B. Algorithm Experiment

To facilitate comparative experiments, we conducted experiments with the pre-collected indoor dataset. The experiments simulated the historical run and collaborative perception of

two PCPs. The vehicle with the green trajectory, referred to as vehicle A, represents the ego vehicle, while vehicle B with the blue trajectory shares early cooperation data to vehicle A.

For the experimental setup, we choose to compare our method with directly transmitting (DT) raw point clouds and performing voxel down-sampling (VD). Voxel down-sampling is an algorithm for filtering point clouds based on voxelization. Generally, after dividing the grids, it takes the average of all points in each grid and replaces them. This is a commonly used method for early cooperation in cases where point clouds collected by multi-line LiDARs are very dense. The voxel size in voxel down-sampling is set to 5cm in experiments, consistent with the voxelization grid size in our algorithm.

Furthermore, it should be noted that our proposed algorithm has the advantage of naturally eliminating points that overlap with the base map, which may not be the case for the other two algorithms. To ensure a fair comparison, we also conducted tests on the other two algorithms after removing overlapping points with the base map (ROM).

To conduct a comprehensive comparison, we choose the following experimental metrics: the frames per second (FPS) of our front-end program, the average size of a single point cloud data (ASP) sent from vehicle B during collaboration, the total number of point clouds (TNP) transmitted during the entire run, and the average bandwidth occupancy (ABO) for transmitting perception data throughout the entire run.

TABLE I
PERFORMANCE AND COMMUNICATION TESTS ON SELECTED ALGORITHMS

Method	FPS	ASP (KB)	TNP	ABO (Mbps)
DT	18.72	673.15	5205	52.59
DT + ROM	17.20	77.98	5205	6.09
VD	17.94	204.39	5205	15.97
VD + ROM	16.36	32.72	5205	2.56
Ours	14.85	15.71	1158	0.27

As can be seen, our algorithm has a significant advantage in terms of communication. This is primarily because a vehicle only transmits points within the demanded area (in our experiments, this value is set to a 20m * 20m area around the vehicle) to the corresponding vehicle. The visual range of the two vehicles cannot always cover each other's demand areas. Therefore, in many cases, data exchange does not occur, resulting in notable bandwidth savings. The advantage of our algorithm becomes more evident in scenarios with limited or large-scale communication. For instance, in an ideal network environment with only 10Mbps of bidirectional bandwidth, the best basic algorithm can accommodate up to 4 concurrently running PCPs, while our algorithm can support up to 38 PCPs, making large-scale testing in such scenarios possible.

The performance loss from our algorithm can be observed in terms of FPS, but in a real run, we only need to ensure that the algorithm's FPS is higher than the sampling rate of the LiDAR (usually 10Hz) to avoid point cloud data loss. Considering the significant improvement at the communication level, this performance loss is completely acceptable.

C. Ablation Experiment

In our cooperative perception algorithm, we apply additional voxelization three times. When calculating the visual range (CVR), we perform real-time voxelization on the point cloud collected by vehicle A, and use the center point of the voxelized grid to replace all points in the grid before performing the Bresenham algorithm traversal. When vehicle B is removing points outside the blind zone (RBZ) from the collected point cloud, we perform voxelization on it, then judge whether the voxelized grid overlaps with the blind zone grid, changing the point-to-grid traversal to a grid-to-grid traversal. Similarly, when vehicle B is removing points overlapping with the base map (ROM), we perform voxelization on the real-time point cloud and compare it with base map grids obtained in the previous process, changing the point-to-point traversal to a grid-to-grid traversal.

In order to verify the impact of voxelization on our algorithm, we separately remove the voxelization methods from the above methods, and record the average running time (AT) of CVR, RBZ and ROM on a single point cloud respectively, the average size of the transmitted point clouds (ASP), and the total number of transmitted point clouds (TNP) under the corresponding settings.

Specifically, we conduct ablation experiments under five different settings. These settings are: enabling voxelization in all algorithms (EVA), disabling voxelization in CVR (DVCVR), disabling voxelization in RBZ (DVRBZ), disabling voxelization in ROM (DVROM), and disabling voxelization in all cases (DVA). The experimental results are as follows:

TABLE II
RESULTS OF ABLATION EXPERIMENT

Method	AT of CVR (ms)	AT of RBZ (ms)	AT of ROM (ms)	ASP (KB)	TNP
EVA	5.50	4.67	4.08	15.71	1158
DVCVR	184.97	4.75	4.12	15.40	1156
DVRBZ	5.54	38.84	4.37	15.71	1158
DVROM	5.41	4.79	403.44	16.21	1160
DVA	182.27	37.60	404.05	15.86	1158

From the results, we can determine the necessity of voxelization in our algorithm. Firstly, in real-time scenarios, considering the 10Hz sampling frequency of the LiDAR, the maximum tolerable computation latency is 100ms. Without enabling voxelization, the only option that meets the requirement is DVRBZ. However, it compares points with grids, and voxelization is performed within the same coordinate system and with the same grid size c, so it does not result in any changes in the amount of transmitted information. Additionally, we can observe that DVCVR actually leads to fewer transmitted point clouds. This is because the point-by-point traversal includes some boundary cases in the calculation, making the computed blind zone more accurate. As for DVROM, some points with distances close to the preset threshold are preserved due to the change from point-to-point to grid-to-grid, thus transmitting more information.

Under the combined effect of the three voxelizations, the number of point clouds transmitted by the EVA and the DVA are precisely aligned. This, to some extent, indicates that the errors introduced by voxelization can be considered negligible. Specifically, EVA results in an average information loss of 0.95% but saves 97.72% of computation time, which is completely acceptable in a real-time environment.

V. Conclusion and Future Work

This paper presents PCP, a collaborative perception system for autonomous driving algorithms, serving as a low-cost transitional solution between software simulation and real vehicle road testing. The system utilizes ROS for driver-level hardware interaction, including raw data collection and vehicle control, etc. On the upper level, we have developed a highly customizable front-end for data parsing and visualization. To provide a solution for real-time collaborative perception, we apply an efficient voxelized grid-based early cooperation algorithm to facilitate data sharing among vehicles while maintaining the LiDAR sampling frequency.

Considering the existing limitations of the system, our future work includes the following:

- Real road testing: although our system has performed data collection and algorithm validation in indoor and outdoor environments, running the system on roads with real vehicles remains a challenge. This objective involves road safety and legal compliance considerations, and we will actively work towards addressing these issues to achieve a closer alignment with real road testing.
- Communication environment: the early cooperation algorithm in this paper is tested on the dataset collected by the system, without extensive focus on communication bottlenecks in large-scale scenarios. To validate the system's performance limits under communication constraints, we will conduct large-scale communication scenario tests to verify the system's communication capabilities.
- Robot2Real: similar to the "reality gap" in simulators, we believe there is a "device gap" between existing devices and real vehicles. The robots run in a realistic environment, solving challenges such as lighting and dynamics that simulators struggle with. However, there are differences in the size of the robot and real vehicles, as well as variations in the positions of sensors, and pedestrians' and vehicles' responses may be entirely different. This means that algorithms tested within PCP also face challenges in deployment. Designing "Robot to Real Vehicle", or Robot2Real as we call it, to bridge the gap between the system and real vehicles will be a focus of our next phase of work.

Acknowledgment

The design of PCP would not have been possible without the Autolabor robot platform and the integrated Autolabor OS shared by Autolabor. Its rich resources and modular ROS components have greatly accelerated our development. The authors would like to appreciate their support.

References

[1] Q. Chen, S. Tang, Q. Yang, and S. Fu, "Cooper: Cooperative perception for connected autonomous vehicles based on 3d point clouds," in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 514–524.

[2] E. Arnold, M. Dianati, R. de Temple, and S. Fallah, "Cooperative perception for 3d object detection in driving scenarios using infrastructure sensors," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 1852–1864, 2020.

[3] X. Zhang, A. Zhang, J. Sun, X. Zhu, Y. E. Guo, F. Qian, and Z. M. Mao, "Emp: Edge-assisted multi-vehicle perception," in *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*, 2021, pp. 545–558.

[4] R. Xu, H. Xiang, Z. Tu, X. Xia, M.-H. Yang, and J. Ma, "V2x-vit: Vehicle-to-everything cooperative perception with vision transformer," in *European conference on computer vision*. Springer, 2022, pp. 107–124.

[5] Q. Chen, X. Ma, S. Tang, J. Guo, Q. Yang, and S. Fu, "F-cooper: Feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 2019, pp. 88–100.

[6] H. Yu, Y. Luo, M. Shu, Y. Huo, Z. Yang, Y. Shi, Z. Guo, H. Li, X. Hu, J. Yuan *et al.*, "Dair-v2x: A large-scale dataset for vehicle-infrastructure cooperative 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 21 361–21 370.

[7] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, F. Golemo, C. Atkeson, D. Fox, K. Goldberg, J. Leonard *et al.*, "Sim2real in robotics and automation: Applications and challenges," *IEEE transactions on automation science and engineering*, vol. 18, no. 2, pp. 398–400, 2021.

[8] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra, "Sim2real predictivity: Does evaluation in simulation predict real-world performance?" *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6670–6677, 2020.

[9] R. Molina-Masegosa, M. Sepulcre, J. Gozalvez, F. Berens, and V. Martinez, "Empirical models for the realistic generation of cooperative awareness messages in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 5, pp. 5713–5717, 2020.

[10] S. Aoki, T. Higuchi, and O. Altintas, "Cooperative perception with deep reinforcement learning for connected vehicles," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 328–334.

[11] G. Luo, H. Zhou, N. Cheng, Q. Yuan, J. Li, F. Yang, and X. Shen, "Software-defined cooperative data sharing in edge computing assisted 5g-vanet," *IEEE Transactions on Mobile Computing*, vol. 20, no. 3, pp. 1212–1229, 2021.

[12] R. Yu, D. Yang, and H. Zhang, "Edge-assisted collaborative perception in autonomous driving: A reflection on communication design," in *2021 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2021, pp. 371–375.

[13] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3d: A modern library for 3d data processing," *arXiv preprint arXiv:1801.09847*, 2018.

[14] J. Fang, D. Zhou, X. Song, and L. Zhang, "Mapfusion: A general framework for 3d object detection with hdmaps," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 3406–3413.

[15] Y. Huang, J. Zhou, X. Li, Z. Dong, J. Xiao, S. Wang, and H. Zhang, "Menet: Map-enhanced 3d object detection in bird's-eye view for lidar point clouds," *International Journal of Applied Earth Observation and Geoinformation*, vol. 120, p. 103337, 2023.

[16] A. Koval, C. Kanellakis, and G. Nikolakopoulos, "Evaluation of lidar-based 3d slam algorithms in subt environment," *IFAC-PapersOnLine*, vol. 55, no. 38, pp. 126–131, 2022.

[17] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 1271–1278.

[18] J. E. Bresenham, "Algorithm for computer control of a digital plotter," in *Seminal graphics: pioneering efforts that shaped the field*, 1998, pp. 1–6.

[19] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.