Hetecooper: Feature Collaboration Graph for Heterogeneous Collaborative Perception

Congzhang Shao[1], Guiyang Luo[1,✉], Quan Yuan[1,✉], Yifu Chen[1], Yilin Liu[1], Kexin Gong[1], and Jinglin Li[1]

State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
`{shaocongzhang,luoguiyang,chenyifu,yuanquan,liuyilin10}@bupt.edu.cn`,
`suroooprise@gmail.com`, `jlli@bupt.edu.cn`

Abstract. Collaborative perception effectively expands the perception range of agents by sharing perceptual information, and it addresses the occlusion problem in single-vehicle perception. Most of the existing works are based on the assumption of perception model homogeneity. However, in actual collaboration scenarios, agents use different perception model architectures, which leads to differences in the size, number of channels and semantic space of intermediate features shared to collaborators, bringing challenges to collaboration. We introduce Hetecooper, a collaborative perception framework for scenarios with heterogeneous perception models. To model the correlation between heterogeneous features, we construct the feature collaboration graph, which completely preserves the semantic information and spatial information of features. Furthermore, a message passing mechanism based on graph transformer is designed to transfer feature messages in the feature collaboration graph. Firstly, the number of node channels and the semantic space are unified by the semantic mapper. Then, the feature information is aggregated by the edge weight guided attention, and finally the fusion of heterogeneous features is realized. Test results demonstrate that our method achieves superior performance in both model homogeneity and heterogeneity scenarios, and also has good scalability to the change of feature size.

Keywords: Collaborative perception · Autonomous driving · Graph transformer

1 Introduction

Collaborative perception has garnered significant attention in recent years. By sharing perception information among agents, it can significantly expand the perception range of agents. In the field of autonomous driving, it can effectively mitigate the occlusion and blind spot issues that are difficult to solve in single vehicle perception, and provide crucial support for safer driving decisions.

✉: Corresponding author.

According to the type of perceptual information shared among agents, collaborative perception can be divided into early fusion [16, 21, 43], which involves sharing initial sensor sensing data. Intermediate fusion [4, 8, 13, 18, 29, 35, 37], involves sharing intermediate features extracted from perceptual models. Late fusion [24,32], involves sharing model detection results. Among these, intermediate fusion has gradually become the focus of attention due to the more favorable performance-bandwidth trade-off.

Existing work on collaborative perception largely assumes the isomorphic of perception models. However, in collaborative perception, the heterogeneity of perception models used by different agents is an issue that cannot be ignored. Particularly in intermediate fusion methods, when the architectures of perception models differ, the dimensions, number of channels, and semantic spaces of intermediate features shared between agents also vary. How to seamlessly fuse heterogeneous features remains a challenge. Existing methods [15, 34] align heterogeneous features by first matching feature dimensions through interpolation/convolution methods, then aligning feature spaces using attention mechanisms or fine-tuning encoders, and finally integrating the features within a unified semantic space using the original fusion module. However, the interpolation/convolution methods used to align feature sizes approximate the features, leading to information loss during the fusion process.

To address the issue of information loss in heterogeneous feature fusion, we introduce Hetecooper, a framework for heterogeneous collaboration based on intermediate fusion. The graph structure is leveraged to model the correlation between heterogeneous features: The feature points are used as graph nodes, with edges established between nodes that have spatial correlation, and this spatial correlation is quantified as the edge weight to construct the feature collaboration graph. Furthermore, a method based on the graph transformer is designed to transfer feature messages within the feature collaboration graph: First, the semantic mapper is used to unify the channel number and semantic space of graph nodes. Then, attention is directed to aggregate the information of features by utilizing edge weights, that is, the fusion of heterogeneous features is realized.

Experimental results indicate that Hetecooper achieves optimal performance in the scenarios of model homogeneity and model heterogeneity. Moreover, the trained collaborative model can also adapt well to small changes in feature size/number of channels without adaptation. Our contributions are:

- We propose a collaborative perception framework Hetecooper, suitable for scenarios with heterogeneous perception models. By constructing a feature collaboration graph to model the correlations between heterogeneous features, Hetecooper can integrally preserves the semantic information and spatial information between features, achieving seamless fusion.
- To transmit messages in the feature collaboration graph, we design a graph transformer method. Use edge weights containing spatial information to guide attention weights, nodes can assign higher weights to information that is more beneficial for collaboration.

- The design of the semantic mapper and the construction method of the feature collaboration graph allow Hetecooper to automatically adapt to minor changes in feature size and channel number without the need for adaptation, providing good scalability.
- Experimental results demonstrated that Hetecooper exhibits excellent performance in the scenarios of model homogeneity and model heterogeneity. Moreover, it shows good scalability to minor variations in feature.

2 Related Work

Collaborative Perception. Collaborative perception has gained extensive attention in recent years. By sharing perception information such as images and lidar point clouds among agents, it expands the perception range of the agents, thereby providing support for safer driving decisions. Early work on multi-agent collaborative perception was done by sharing initial sensing information [16, 17, 21, 23, 25, 43] (early fusion) or detection results [24, 32] (late fusion) between agents to achieve collaboration. However, in these two approaches, the early fusion will bring large bandwidth consumption, and the late fusion has low accuracy. Recently work [8, 11–14, 19, 28, 29, 35, 41] shares intermediate feature outputs(intermediate fusion) of perception models between participating collaborative vehicles and fuses features from different agents using attention [1], transformer [27] and other methods, and achieved a good performance-bandwidth trade-off. In this paper, the collaborative perception between heterogeneous agents is realized based on intermediate fusion.
Heterogeneous Agents Collaboration. Several studies have initiated explorations into the issue of heterogeneity in collaborative perception. V2X-ViT [37] addresses the heterogeneity of sensor device architecture by heterogeneous multi-agent self-attention to fuse information across various devices. HM-ViT [31] concentrates on the heterogeneity of perceptual data modalities, designing a 3D graph attention method to learn fusion weights separately for the different correlations between multi-modal features, effectively fusing the feature from camera and lidar modalities. In terms of perceptual model heterogeneity, Xu [34] employs a learnable feature resizer to align features across multiple dimensions and a sparse cross-domain transformer for domain adaptation to fuse heterogeneous features. This paper emphasizes the collaboration of agents when the perception model is heterogeneous. Lu [15] designed a novel extensible collaborative perception framework to align features of collaborators into a unified feature space for collaboration. However, the aforementioned methods cause a loss of feature information due to the interpolation/convolution techniques used when merging features of different sizes. This paper achieves lossless fusion of heterogeneous features by constructing a feature collaboration graph.
Graph Transformer. The transformer's exceptional performance in natural language processing and vision has led to several attempts to apply transformer mechanisms to graph neural networks [3, 26, 30, 42]. Dwivedi et al. [6], Kreuzer et al. [9], and GraphiT [20] use Laplacian Eigenvectors and positive definite

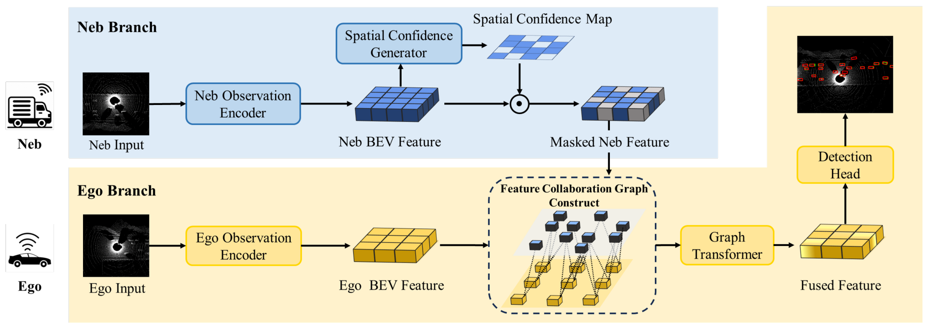


Fig. 1: Framework of Hetecooper. Comprises five stages: observation encoding, feature sharing, feature collaboration graph construction, feature fusion, and detection output. Dashed lines connect the feature points of neb (blue squares) and ego (yellow squares) that exists spatial correlations.

kernels on graph adjacency matrices, respectively, as the positional encoding of nodes. This improves the efficiency of message passing between graph nodes. Rampašek et al. [22] propose a method to build a general, powerful, and scalable graph transformer, they highlight that the key to designing an efficient graph transformer method is to distinguish the position encoding of nodes and implement an efficient message passing mechanism. In this paper, we design an improved graph transformer method that assigns position codes containing spatial information to nodes and uses edge weights to guide message passing, thereby achieving efficient fusion of heterogeneous features.

3 Method

3.1 Problem Statement

The observational information of the N collaborating agents and the corresponding perception supervision labels are denoted as $\mathcal{O}_i$ and $\mathcal{Y}_i$, $i = 1, 2, ..., N$, respectively. Our main goal in collaborative perception is to optimize the parameters in such a way that each agent can achieve the maximum perception performance:

$$\arg\min_{\theta_i} \sum_{i=1}^{N} \mathcal{L}\left(\mathcal{H}_i\left(\mathcal{O}_i, \{\mathcal{I}_{j\longrightarrow i}\}; \theta_i\right), \mathcal{Y}_i\right), where\ j \in \mathcal{N}_i \tag{1}$$

where $\mathcal{N}_i$ is the set of all collaborators of agent i, $\mathcal{H}_i(\cdot)$ represents the perception model utilized by agents, $\mathcal{I}_{j\longrightarrow i} \subset \mathcal{F}_j$ is the message that agent j send to agent i, $\mathcal{F}_j \in \mathbb{R}^{C_j \times H_j \times W_j}$ is the feature extracted from the perception model of agent j. $\mathcal{F}_i$ varies in parameters, sizes, the number of channels, and semantic space.

3.2 Analysis: Collaboration Among Heterogeneous Agents

When the agents use heterogeneous perception models, the size, number of channels and semantic space of shared intermediate features are also different. At this

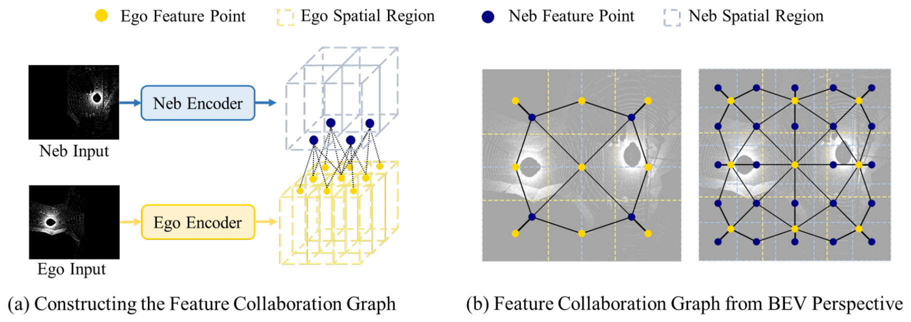


Fig. 2: Schematic diagram of feature collaboration graph. (a) shows the construction process of the feature collaboration graph, (b) shows the feature collaboration graph from the BEV perspective under two different parameter settings, with the background being the point cloud observation data.

point, the challenge of intermediate fusion lies in how to precisely and efficiently fuse heterogeneous intermediate features. A simple approach is to first align the channel number and semantic space of intermediate features by 1x1 convolution, and then align the feature size by sampling/convolution. After unify the feature representation, use existing homogeneous collaborative methods such as [8, 37], etc to fuse feature. However, an obvious problem with this approach is that the sampling/convolution process approximately computes the heterogeneous features and loses feature information.

To address this issue, we conduct an in-depth analysis of the physical significance of intermediate features. The observation encoders [10,39,44] extract BEV (Bird's Eye View) features through the following process: First, use voxel grids of the same size to partition the point cloud observation data. Then, a feature representation is extracted within each grid, and these features are combined according to their spatial relationships to obtain the 3D features $\mathcal{F} \in R^{C_0 \times Z \times H \times W}$. Finally, the features are concatenated along the Z-dimension, which is perpendicular to the ground, to obtain the shared BEV features $\mathcal{F}_{bev} \in R^{(C_0 * Z) \times H \times W}$. Hence, each feature point in the intermediate features corresponds to information from a specific pillar region in the real world.

When the collaborating agents use heterogeneous perception models, the voxel grid sizes of the ego agent and the collaborating agents (neb) differ, leading to different BEV feature representations for the same real-world region. We analyze the real-world region, ego features, and neb features together, as shown in Fig. 2. In the figure, the yellow and blue points represent the feature points in the ego and neb BEV features, respectively. The dashed columns in Fig. 2a and the dashed grids in Fig. 2b represent the entire pillar regions and their bases corresponding to the feature points, respectively. Black lines connect the feature points whose corresponding pillar regions intersect. It is evident that the pillar regions corresponding to the feature points of the ego and neb intersect but do not completely overlap, indicating a complex spatial correlation.

We use a graph structure to describe this relationship: feature points are treated as graph nodes, and edges are established between ego and neb nodes whose corresponding pillar regions intersect, constructing a feature collaboration graph. Subsequently, the graph transformer method is used to propagate and aggregate feature messages within the feature collaboration graph, achieving the fusion of heterogeneous features. Since the initial feature points are directly used as graph nodes when constructing the feature collaboration graph, this method ensures no information loss during the fusion process. The details of graph construction and Graph Transformer fusion will be introduced in Sec. 3.3.

3.3 Framework

The heterogeneous collaborative perception framework Hetecooper proposed in this paper is illustrated in Fig. 1, including five stages: observation information encoding, feature sharing, feature collaboration graph construction, feature fusion, and detection output. Initially, the agent uses an observation encoder to extract BEV features. Subsequently, leveraging a spatial confidence map [8], it filters out reliable features to share with collaborators. Once feature sharing is complete, the agent constructs a feature collaboration graph based on the spatial relationships between its own features and those of the collabors. The graph transformer is then used to propagate the collaborators' feature information to itself, achieving the fusion of heterogeneous features. Finally, the fused features are input into the detection head to obtain detection outputs, thus completing one cycle of the heterogeneous collaborative perception process.

Observation Encoding. The agent extracts the BEV feature representation from the initial sensor data $\mathcal{O}_i$ by using the observation encoder $\mathcal{H}_{enc_i}(\cdot)$. Firstly, voxelization is applied to the point cloud, followed by the extraction of feature representations within each voxel grid. Then, the feature representations are connected to form complete BEV feature representations. Subsequently, further refinement is achieved through a 2D backbone network (such as ResNet [7], etc.), resulting in the final BEV feature representations $\mathcal{F}_i = \mathcal{H}_{enc_i}(\mathcal{O}_i) \in \mathbb{R}^{C_i \times H_i \times W_i}$. The $\mathcal{H}_{enc_i}(\cdot)$ can be any existing observation encoder framework, such as PointPillar [10], SECOND [39], etc.

Feature Sharing. Heterogeneous agents share intermediate features. To enhance the efficiency of heterogeneous feature fusion, agents utilize a spatial confidence map generator $\mathcal{H}_{conf_i}(\cdot)$ (sharing parameters with the Detection Head) to filter critical information from the BEV features $\mathcal{F}_i$. First, use $\mathcal{H}_{conf_i}(\cdot)$ to compute a spatial confidence map [8] $\mathcal{C}_i = \mathcal{H}_{enc_i}(\mathcal{F}_i) \in [0,1]^{H_i \times W_i}$ (regions with confidence below a threshold are set to 0, and those above the threshold are set to 1). Then, perform element-wise multiplication between $\mathcal{C}_i$ and feature $\mathcal{F}_i$ to obtain key information $\mathcal{I}_i = \mathcal{C}_i \odot \mathcal{F}_i \in \mathbb{R}^{C_i \times H_i \times W_i}$ for transmission and broadcast it to collaborators, thus completing the feature sharing process.

Feature Collaboration Graph Construct. Establish a feature collaboration graph between ego features $\mathcal{F}_i$ and information $\mathcal{I}_{j\longrightarrow i} \subset \mathcal{F}_j$ received from neighboring (neb) agents based on spatial relationships between feature points. First, according to the relative position relationship between ego and neb, warp the neb information $\mathcal{I}_{j\longrightarrow i}$ into a view centered on ego. Subsequently, compute the base range of the pillar region $\mathcal{M}(\mathcal{P})$ corresponding to feature points $\mathcal{P}_m^i \in \mathcal{F}_i$ in $\mathcal{F}_i$ and feature points $\mathcal{P}_n^j \in \mathcal{I}_{j\longrightarrow i}$ in $\mathcal{I}_{j\longrightarrow i}$ based on model parameter settings (see detailed process in Sec. 6.1 in Appendix). Through $\mathcal{M}(\mathcal{P})$, we can determine the spatial correlations between feature points, thereby constructing a feature collaboration graph, specific method is as follows:

If feature points $\mathcal{P}_n^j$ from neb and feature points $\mathcal{P}_m^i$ from ego correspond to overlapping base areas of pillar regions or their center point distance is less than threshold, that is, satisfying either of the following two conditions:

$$S\left(\mathcal{M}\left(\mathcal{P}_m^i\right) \cap \mathcal{M}\left(\mathcal{P}_n^j\right)\right) > 0 \tag{2}$$

$$dis\left(\mathcal{M}\left(\mathcal{P}_m^i\right), \mathcal{M}\left(\mathcal{P}_n^j\right)\right) < D \tag{3}$$

then ego node $\mathcal{P}_m^i$ and neb node $\mathcal{P}_n^j$ is called to have a spatial correlation, and is added to the node set $\mathcal{V}$. And edge $e_{mn}\left(\mathcal{P}_m^i, \mathcal{P}_n^j\right)$ is added to the edge set $\mathcal{E}$.

Where $\cap$ represents the overlapping area between the base regions of two cylindrical areas, $S(\cdot)$ represents the calculation of the area of the spatial region base, D is the maximum distance threshold, and $dis(\cdot)$ represents the calculation of the distance between the center points of the base regions of two cylindrical areas. The setting of criterion Eq. (3) is aimed at establishing correlations between ego feature points and a broader range of neb feature points, providing additional reference information for collaboration.

Furthermore, we refer to Eq. (2) as spatial correlation and Eq. (3) as proximity distance, quantifying each of them separately to assign weights to the edges in feature collaboration graph.

Spatial correlation: The ratio of the overlapping area of the base regions of cylindrical areas corresponding to ego and neb nodes to the area of the base region corresponding to ego nodes.

$$spc_{mn} = \frac{S\left(\mathcal{M}\left(\mathcal{P}_m^i\right) \cap \mathcal{M}\left(\mathcal{P}_n^j\right)\right)}{S\left(\mathcal{M}\left(\mathcal{P}_m^i\right)\right)} \tag{4}$$

where $S(\cdot)$ represents the area of the bottom surface of the calculated spatial region, $\mathcal{M}(\cdot)$ denotes the corresponding spatial region of nodes, and $\cap$ signifies the overlap between the two regions.

Distance proximity: The components of the distance between the center points of the pillar regions corresponding to ego and neb nodes along the two coordinate axes:

$$dx_{mn} = abs\left(x_n - x_m\right) \tag{5}$$

$$dy_{mn} = abs\left(y_n - y_m\right) \tag{6}$$

where (x_m, y_m) and (x_n, y_n) represent the coordinates of the center points of the spatial regions corresponding to nodes $\mathcal{P}_m^i$ and $\mathcal{P}_n^j$, respectively. The coordinate system takes the ego as the origin, and the front and right sides of the ego are positive for the y-axis and square for the x-axis, respectively. Here, the two components of the distance are added to the edge weights respectively in order to better distinguish the relative positions of the nodes in the corresponding spatial regions.

Finally, by combining spatial similarity and distance proximity, we get the weights of edge e_{mn}:

$$e_{mn} = (spc_{mn}, dx_{mn}, dy_{mn}) \tag{7}$$

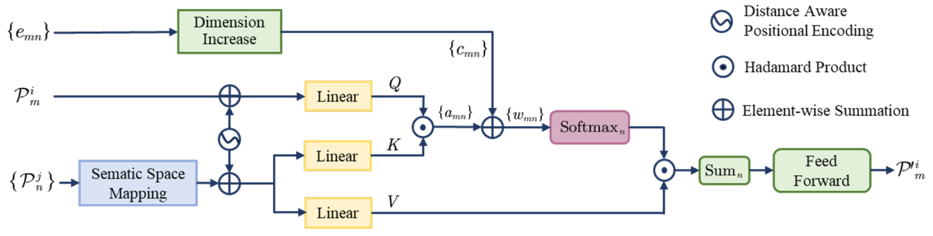


Fig. 3: Detailed design of the graph transformer based fusion module

Feature Fusion. The Graph Transformer has gained popularity due to its exceptional ability to handle complex patterns. In light of this, we have designed a fusion module $\mathcal{H}_{fuse_i}(\cdot)$ based on the graph transformer to pass and aggregate messages in feature collaboration graph. As per the analysis in [22], the key to designing an effective graph transformer lies in distinguishing the positional encoding of nodes and ensuring an efficient message passing mechanism. To achieve this, we assign a positional encoding to the nodes that is related to the spatial distance, and then guide the nodes' attention through the edge weights, assigning higher weights to nodes that contribute more beneficial information to collaboration, thereby enhancing the efficiency of feature message propagation. Fig. 3 illustrates the detailed structure of the fusion module $\mathcal{H}_{fuse_i}(\cdot)$.

Node Semantic Space Unify. Ego nodes and neb nodes originate from different features, exhibiting differences in channel count and semantic space. Initially, we align the feature representation of neb nodes with those of ego nodes using a semantic mapping module $\mathcal{H}_{map_i}(\cdot)$. $\mathcal{H}_{map_i}(\cdot)$ is designed with reference to [2] and requires separate adaptation for the Observation Encoder of each architecture. Additionally, before the features are input into $\mathcal{H}_{map_i}(\cdot)$, they pass through a channel aligner module to automatically adapt to minor variations in the number of channels. Detailed design is in Sec. 6.2 in Appendix.

Node Positional Encoding. As analyzed in Sec. 3.2, each feature point in the BEV features corresponds to information within a certain pillar region in the real world. We introduce position encodings related to spatial information for the nodes $\mathcal{P}$ to distinguish between different nodes. If the number of channels for node $\mathcal{P}$ is C, and the distance from the center of the base of the corresponding pillar region to the center of the ego agent is $D_{\mathcal{P}}$, then its positional encoding is calculated as follows:

$$PE\left(D_{\mathcal{P}}\right)=\begin{cases}\sin\left(D_{\mathcal{P}}/10000^{\frac{2c}{C}}\right), c=2k\\ \cos\left(D_{\mathcal{P}}/10000^{\frac{2c}{C}}\right), c=2k+1\end{cases} \tag{8}$$

Subsequently, we process $PE\left(D_{\mathcal{P}}\right)$ through a fully connected layer f and add it to the node $\mathcal{P}$ to obtain the node representation with distance information.

$$\mathcal{P}=\mathcal{P}+f\left(PE\left(D_{\mathcal{P}}\right)\right) \tag{9}$$

Edge weights guide attention. Aggregating neb node information into the ego node. For an ego node $\mathcal{P}_m^i$, if the set of its neighboring nodes in the feature collaboration graph is denoted as $\mathcal{A}_m$, then for each $\mathcal{P}_n^j\in\mathcal{A}_m$, we first calculate the initial attention weight:

$$a_{mn}=\frac{W_Q\mathcal{P}_m^i\odot W_K\mathcal{P}_n^j}{\sqrt{C}} \tag{10}$$

where $W_Q\in\mathbb{R}^{C\times C}$ and $W_K\in\mathbb{R}^{C\times C}$ are used to generate the query and key vectors, respectively. C denotes the number of channels in the node representation, and $\odot$ signifies the execution of the Hadamard Product on the vector.

Then, we compute the corrective term of attention derived from the edge weights $e_{mn}=(spc_{mn},dx_{mn},dy_{mn})$:

$$c_{mn}=\frac{(spc_{mn}+1)}{2}\cdot\left(PE\left(dx_{mn}\right)||PE\left(dy_{mn}\right)\right) \tag{11}$$

where $PE(\cdot)$ denotes the positional encoding, calculated in the same way as in Eq. (8), with the value of C set to half the number of feature channels of the node. $||$ denotes the concatenation of two vectors. By concatenating the vectors obtained from the cosine and sine decomposition of the proximity in distance dx_{mn} and dy_{mn}, we obtain a vector with the same dimensions as the nodes. This vector is then multiplied by the spatial correlation spc_{mn} plus 1 and divided by 2, resulting in the attention weight correction term c_{mn}. Adding 1 to spc_{mn} is to prevent the inability to utilize edge weight information when spc_{mn} is 0. Dividing by 2 is to normalize the value after adding 1 to spc_{mn}.

After passing through the fully connected layer g, the attention modifier c_{mn} is added to a_{mn}, yielding the final attention weight w_{mn}:

$$w_{mn}=a_{mn}+g\left(c_{mn}\right) \tag{12}$$

Subsequently, the value vectors of all ego's adjacent nodes $\mathcal{P}_n^j\in\mathcal{A}_m$ are multiplied by the weights $w'_{mn}=\operatorname{soft\,max}_n\left(w_{mn}\right)$, obtained from w_{mn} by softmax normalization, and obtain the ego node that aggregates the neb information:

$$\mathcal{P}\prime_m^i=\sum_{n\in\mathcal{A}_m}w'_{mn}W_V\mathcal{P}_n^j \tag{13}$$

where $W_V \in \mathbb{R}^{C\times C}$ are used to generate the value vector.

Finally, the updated node representation $\mathcal{P}_m^i$ is reassembled according to its index in the ego feature $\mathcal{F}_i$, that is, the ego feature $\mathcal{F}_{ij}$ after fusing neb information $\mathcal{I}_{j\longrightarrow i}$ is obtained.

Multi-scale collaboration. To capture the correlation between the voxel-level and the target contour level within the heterogeneous features, we integrate neb information across various scales. We employ the Split-atten method [40] to combine the fusion results $\mathcal{F}_{ij}^s$, where $s = 1, 2, .., S$, from different scales to obtain $\mathcal{F}'_{ij} = \text{Split}\left(\left\{\mathcal{F}_{ij}^s\right\}_{s=1}^S\right)$. The detailed process is in Sec. 6.4 in Appendix.

Finally, we perform cross-attention between the ego and each neb's fused feature $\mathcal{F}'_{ij}$, and by applying a feed forward layer, we derive the ego feature that integrates all neb information:

$$\mathcal{F}'_i = \text{FeedForward}\left(\text{ATTEN}\left(\mathcal{F}'_{ij}\right)\right), j \in \mathcal{N}_i \tag{14}$$

Detection Output. The detection results $\mathcal{D}_i = \mathcal{H}_{head_i}\left(\mathcal{F}'_i\right) \in \mathbb{R}^{H\times W\times 7}$ are decoded from the fused features $\mathcal{F}'_i$. Here, $\mathcal{D}_i = (cls, x, y, z, len, \cos\theta, \sin\theta)$ represents the rotation box with categories, the elements in the parentheses denote the class, position coordinates, target length, and rotation angle, respectively.

3.4 Training

The training of Hetecooper is divided into two steps. In the first step, the model is set to be homogeneous, learning all model parameters. In the second step, the model is set to be heterogeneous, keeping the parameters of observation encoder $\mathcal{H}_{enc_i}(\cdot)$ unchanged, while adapting a set of parameters of fusion module $\mathcal{H}_{fuse_i}(\cdot)$ and detection head $\mathcal{H}_{head_i}(\cdot)$ for each architecture. Both training steps use the loss between the detection outputs and the labels for supervised training.

$$L = \sum_{i=1}^{N} L_{\det}\left(\mathcal{D}_i, \mathcal{Y}_i\right) \tag{15}$$

where $\mathcal{D}_i$ is the detection output, $\mathcal{Y}_i$ presents the label of the observation information $\mathcal{O}_i$ corresponding to agent i, and $L_{\det}$ denotes the detection loss.

4 Experiment

4.1 Dateset

OPV2V. OPV2V [38] is a large-scale open dataset for perception with V2V communication. By utilizing a cooperative driving co-simulation framework named OpenCDA [33] and CARLA [5] simulator, it collect divergent scenes with various numbers of connected vehicles. OPV2V has 73 scenes, 6 road types, 9 cities, 12K frames of LiDAR point clouds , 230K annotated 3D bounding box.
V2V4Real. Test results on V2V4Real [36] are presented in Sec. 6.6 in Appendix.

Table 1: Comparison when architecture of observation encoders are different. The terms on the left and right sides of the "-" represent the names of the observation encoders used for the ego and neb, respectively. Results are reported in AP@0.50/AP@0.70. Since the performance of the collaboration method is related to the proportion of heterogeneous agents in the scenario, we uniformly set the maximum number of vehicles in the scenario to 2 to eliminate the influence of unrelated variables.

Observation Encoder	Collaborative Method				
	Hetecooper	HM-ViT	MDPA	Where2comm	Fcooper
SECOND	**0.863**/0.759	0.841/0.647	0.831/0.783	0.838/0.783	0.845/**0.772**
VoxelNet	**0.812/0.731**	0.632/0.483	0.794/0.726	0.794/0.726	0.758/0.685
PointPillar	**0.851**/0.713	0.811/**0.727**	0.816/0.722	0.816/0.722	0.809/0.711
PointPillar - SECOND	**0.851/0.735**	0.808/0.727	0.788/0.673	0.706/0.560	0.679/0.594
PointPillar - VoxelNet	**0.822/0.724**	0.788/0.659	0.675/0.535	0.678/0.549	0.674/0.552
SECOND - PointPillar	**0.859/0.748**	0.833/0.691	0.822/0.722	0.729/0.632	0.708/0.622
Voxelnet - PointPillar	**0.816/0.701**	0.699/0.526	0.789/0.690	0.670/0.568	0.607/0.506

4.2 Experimental Setup

Architecture of observation encoder. The architectures included in this study are PointPillar [10], SECOND [39], and VoxelNet [44]. The default voxel grid sizes in model parameters for these architectures are set to [0.4, 0.4, 0.4], [0.4, 0.4, 0.4], and [0.1, 0.1, 0.1] (in meters), respectively.
Baseline. We compare Hetecooper with HM-ViT [31], MDPA [34](which is applied in Where2comm), Where2comm [8] and F-cooper [4]. The latter two fusion methods are designed under the assumption of model homogeneity and cannot directly fuse heterogeneous features. Therefore, we first use the simple method mentioned in Sec. 3.2 to align the feature representations, and then use the originally designed fusion module to integrate the features from the collaborators, detailed procedures and training methods are in Sec. 6.3 in Appendix.

4.3 Quantitative Analysis

Observation encoder architecture heterogeneity. As shown in Tab. 1, we compare the performance of Hetecooper with other collaborative methods when observation encoder architectures are heterogeneous. It is evident that Hetecooper outperforms the others under conditions of model heterogeneity (lines 4-7), effectively minimizing the impact of heterogeneity on collaboration. This is because Hetecooper constructs the feature collaboration graph by directly using feature points as graph nodes, preserving the complete feature information, whereas other methods approximate the features, leading to information loss. On the other hand, the attention mechanism, guided by edge weights, enhances the efficiency of feature information transfer. And assign higher weights to the information that is more beneficial for collaboration at the nodes.

Besides, under conditions of model homogeneity (lines 1-3), Hetecooper also achieves the best performance at AP@0.5. This is because the feature collaboration graph, when constructed, includes all neb nodes within the distance threshold in the attention scope of the ego node, providing a wealth of reference information for collaboration. Here, we do not compare with HEAL [15], this is because HEAL collaborates in a negotiated common feature space, whereas other methods collaborate directly in the local semantic space, which is unfair.

Table 2: Parameter settings. observation encoder architecture is PointPillar

Parameters	Voxel Size (vx, vy, vz)	Output Channel	Feature Size (C, H, W)
$p1$	0.4, 0.4, 4	256	256, 50, 176
$p2$	0.3, 0.3, 4	256	256, 66, 234
$p3$	0.6, 0.6, 4	256	256, 33, 117
$p4$	0.6, 0.6, 4	272	272, 33, 117
$p5$	0.4, 0.4, 4	272	272, 50, 176

Table 3: Ablation experiment of feature fusion module.

NPE	MS	EGA	AP@0.5	AP@0.7	AP@0.5	AP@0.7
			Homo		Hete	
			0.774	0.586	0.808	0.668
✔			0.805	0.644	0.844	0.692
✔	✔		0.831	**0.721**	0.845	0.725
✔	✔	✔	**0.852**	0.713	**0.851**	**0.735**

Table 4: Extensibility of Hetecooper. Both ego and neb use the PointPillar architecture for observation encoders, and MDPA is based on the Where2comm method. Homo - xx stands for model homogeneity, Hete - xx for model heterogeneity.

Collaborative Method	Parameters									
	AP@0.5					AP@0.7				
	$p1$(ego)	$p2$	$p3$	$p4$	$p5$	$p1$(ego)	$p2$	$p3$	$p4$	$p5$
Homo - Hetecooper	**0.852**	**0.829**	**0.833**	**0.830**	**0.848**	**0.713**	**0.726**	**0.674**	**0.634**	0.693
Homo - MDPA	0.816	0.816	0.789	0.815	0.799	0.722	0.717	0.612	0.660	**0.696**
Hete - Hetecooper	-	**0.869**	**0.586**	0.608	**0.488**	-	**0.701**	**0.406**	0.467	**0.480**
Hete - MDPA	-	0.726	0.544	**0.693**	0.219	-	0.554	0.377	**0.551**	0.160

Extensibility. In order to test the scalability of Hetecooper when there are small differences in the size and number of channels of shared features due to different parameter settings with the same model architecture, we compare Hetecooper with MDPA. During training, the parameters of the ego node were set to $p1$ and adapted with the neb nodes set to $p2$ to obtain the fusion model.

Since the nodes in the feature collaboration graph are directly taken from the initial intermediate features and are independent of feature size, and the design of the channel aligner allows Hetecooper to adapt to any number of feature channels, the trained Hetecooper collaboration model can automatically adapt to minor variations in feature size and channel count without requiring further adaptation. We directly collaborated the fusion model adapted with $p2$ with the neb settings $p3$, $p4$, and $p5$. The results are shown in Tab 4. It is evident that Hetecooper outperforms MDPA when collaborating with neb whose parameters are set as $p2$, $p3$, and $p5$, demonstrating better adaptability to minor variations in size and channels.

Localization Error Robustness. We further evaluated the performance of Hetecooper and the baseline in scenarios with localization errors, as depicted in Fig. 4. As the localization error increases, the performance of all collaboration methods declines. However, Hetecooper consistently outperforms the others.

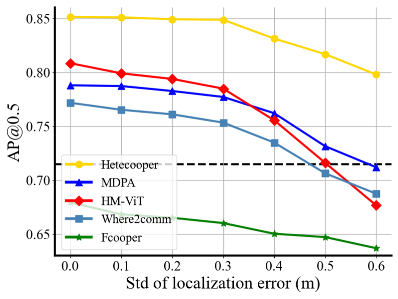


Fig. 4: Robustness to positioning errors. PointPillar and SECOND are used as observation encoders for ego and neb, respectively.

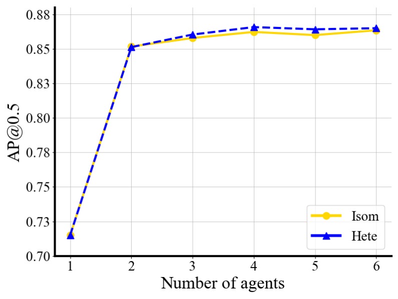


Fig. 5: Visualization of feature collaboration graph. Ego and neb use PointPillar and SECOND observation encoders, respectively.

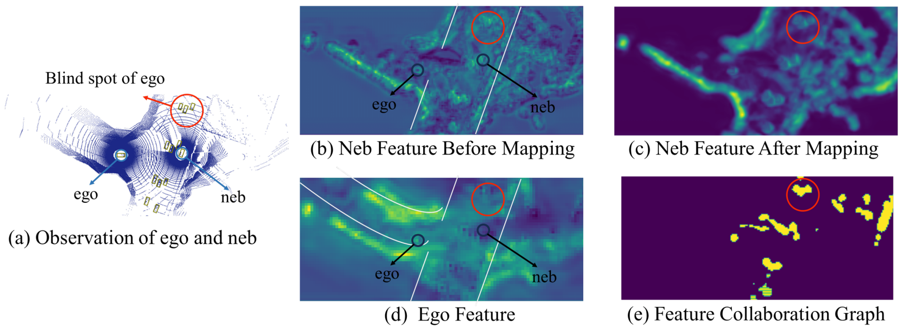

Blind spot of ego
ego
neb
(a) Observation of ego and neb
(b) Neb Feature Before Mapping
(c) Neb Feature After Mapping
(d) Ego Feature
(e) Feature Collaboration Graph

Fig. 6: Visualization of feature collaboration graph. Ego and neb use PointPillar and SECOND observation encoders, respectively.

This is because Hetecooper, when constructing the feature collaboration graph, associates the ego node with all neb nodes within the distance threshold, in the presence of localization errors, Hetecooper can still potentially incorporate the original corresponding neb nodes into the ego node's attention, enabling the ego node to incorporate information that is useful for collaboration.

4.4 Qualitative Analysis

Feature Collaboration Map Visualization. Fig. 6 shows the process of constructing a feature collaboration graph. In the scene, the ego vehicle is moving from the ramp to the main road and cannot see the vehicles in the red circle due to road obstruction. The ego's collaborator neb are on the main road, and the vehicles within the red circle are within the field of its view. Fig. 6b and Fig. 6d display the feature representations taken by the ego and neb, respectively. Fig. 6c presents the representation of neb features after being refined by the semantic mapping module, highlighting that the semantic mapping module can significantly reduces the semantic difference between ego and neb. Upon receiving the neb features, ego creates a collaborative feature graph using the spatial correlation between feature points. We sum up the attention correction terms calculated by Eq. (11) between the ego node and all its neighbors to obtain a one-dimensional matrix and visualize it, as is shown in Fig. 6e. As observed, the feature collaboration graph incorporates the features of vehicles within the red circle into its nodes, effectively filling the blind spots in the ego's field of view.

4.5 Ablation Studies

Ablation Study with Feature Fusion Modules. To test each component's impact in the graph transformer, we integrated node position encoding (NPE), multi-scale fusion (MS), and edge weight-guided attention (EGA) into the graph transformer step by step. We then evaluated their collaborative performance under both model heterogeneity (PointPillar for ego, SECOND for neb) and model homogeneity conditions (PointPillar for both ego and neb), as shown in Tab. 3. All components were found to improve the collaboration effect under model heterogeneity and homogeneity. This is because the positional encoding introduces distance information to the nodes, effectively differentiating between different nodes; multi-scale fusion creating feature collaboration graphs at different scales, richer correlations among heterogeneous features are modeled; edge weight-guided attention allows nodes to assign higher weights to information that is more beneficial for collaboration. These components collectively improve the efficiency of feature message passing and aggregation in the graph transformer, thereby enhancing collaborative performance.

Number of Agents. Fig. 5 displays Hetecooper's performance under both model heterogeneity and homogeneity conditions as the maximum number of vehicles in the scene changes. It's observed that Hetecooper's performance improves as the number of vehicles increases. This improvement is attributed to the fact that more vehicles offer more reference information for collaboration. However, in some instances, detection accuracy declines as the number of vehicles grows. This may be due to the inconsistency in feature representations among different collaborators, which interferes with originally effective information during the fusion process, leading to a decrease in fusion accuracy.

5 Conclusion

This paper is focused on addressing the issue of collaborative perception in scenarios involving heterogeneous perception models, We introduce Hetecooper, a framework for heterogeneous collaborative perception that relies on intermediate fusion. The feature collaboration graph is employed to model the correlation among heterogeneous features, thereby preserving the entirety of the semantic information and spitial information. To improve the efficiency of feature message passing in the feature collaboration graph, we designed an improved Graph Transformer method that uses edge weights to guide attention weights, enabling the lossless and efficient fusion of heterogeneous features. Extensive experiments show the superior performance and good practicability of the proposed method.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China under Grant 2023YFB4301904, in part by the Natural Science Foundation of China under Grant 62102041, Grant 62272053 and Grant 62001054, in part by the Young Elite Scientists Sponsorship Program by China Association for Science and Technology (CAST) under Grant 2022QNRC001, and in part by the Beijing Nova Program under Grant 20230484364.

References

1. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
2. Baldi, P.: Autoencoders, unsupervised learning, and deep architectures. In: Proceedings of ICML workshop on unsupervised and transfer learning. pp. 37–49. JMLR Workshop and Conference Proceedings (2012)
3. Chen, D., O'Bray, L., Borgwardt, K.M.: Structure-aware transformer for graph representation learning. In: International Conference on Machine Learning (2022), `https://api.semanticscholar.org/CorpusID:246634635`
4. Chen, Q., Ma, X., Tang, S., Guo, J., Yang, Q., Fu, S.: 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. pp. 88–100 (2019)
5. Dosovitskiy, A., Ros, G., Codevilla, F., López, A.M., Koltun, V.: Carla: An open urban driving simulator. In: Conference on Robot Learning (2017), `https://api.semanticscholar.org/CorpusID:5550767`
6. Dwivedi, V.P., Bresson, X.: A generalization of transformer networks to graphs. arXiv preprint arXiv:2012.09699 (2020)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
8. Hu, Y., Fang, S., Lei, Z., Zhong, Y., Chen, S.: Where2comm: Communication-efficient collaborative perception via spatial confidence maps. Advances in neural information processing systems **35**, 4874–4886 (2022)
9. Kreuzer, D., Beaini, D., Hamilton, W., Létourneau, V., Tossou, P.: Rethinking graph transformers with spectral attention. Advances in Neural Information Processing Systems **34**, 21618–21629 (2021)
10. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12697–12705 (2019)
11. Lei, Z., Ren, S., Hu, Y., Zhang, W., Chen, S.: Latency-aware collaborative perception. In: European Conference on Computer Vision (2022), `https://api.semanticscholar.org/CorpusID:250627145`
12. Li, Y., Ren, S., Wu, P., Chen, S., Feng, C., Zhang, W.: Learning distilled collaboration graph for multi-agent perception. Advances in Neural Information Processing Systems **34**, 29541–29552 (2021)
13. Liu, Y.C., Tian, J., Glaser, N., Kira, Z.: When2com: Multi-agent perception via communication graph grouping. In: Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition. pp. 4106–4115 (2020)

14. Liu, Y.C., Tian, J., Ma, C.Y., Glaser, N., Kuo, C.W., Kira, Z.: Who2com: Collaborative perception via learnable handshake communication. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 6876–6883. IEEE (2020)
15. Lu, Y., Hu, Y., Zhong, Y., Wang, D., Chen, S., Wang, Y.: An extensible framework for open heterogeneous collaborative perception. arXiv preprint arXiv:2401.13964 (2024)
16. Luo, G., Shao, C., Cheng, N., Zhou, H., Zhang, H., Yuan, Q., Li, J.: Edgecooper: Network-aware cooperative lidar perception for enhanced vehicular awareness. IEEE Journal on Selected Areas in Communications (2023)
17. Luo, G., Shao, C., Cheng, N., Zhou, H., Zhang, H., Yuan, Q., Li, J.: Edgecooper: Network-aware cooperative lidar perception for enhanced vehicular awareness. IEEE Journal on Selected Areas in Communications **42**(1), 207–222 (2024). `https://doi.org/10.1109/JSAC.2023.3322764`
18. Luo, G., Zhang, H., Yuan, Q., Li, J.: Complementarity-enhanced and redundancy-minimized collaboration network for multi-agent perception. Proceedings of the 30th ACM International Conference on Multimedia (2022), `https://api.semanticscholar.org/CorpusID:252782950`
19. Luo, G., Zhang, H., Yuan, Q., Li, J.: Complementarity-enhanced and redundancy-minimized collaboration network for multi-agent perception. In: Proceedings of the 30th ACM International Conference on Multimedia. pp. 3578–3586 (2022)
20. Mialon, G., Chen, D., Selosse, M., Mairal, J.: Graphit: Encoding graph structure in transformers. arXiv preprint arXiv:2106.05667 (2021)
21. Qiu, H., Huang, P., Asavisanu, N., Liu, X., Psounis, K., Govindan, R.: Autocast: Scalable infrastructure-less cooperative perception for distributed collaborative driving. arXiv preprint arXiv:2112.14947 (2021)
22. Rampášek, L., Galkin, M., Dwivedi, V.P., Luu, A.T., Wolf, G., Beaini, D.: Recipe for a general, powerful, scalable graph transformer. Advances in Neural Information Processing Systems **35**, 14501–14515 (2022)
23. Rauch, A., Klanner, F., Rasshofer, R., Dietmayer, K.: Car2x-based perception in a high-level fusion architecture for cooperative perception systems. In: 2012 IEEE Intelligent Vehicles Symposium. pp. 270–275. IEEE (2012)
24. Rawashdeh, Z.Y., Wang, Z.: Collaborative automated driving: A machine learning-based method to enhance the accuracy of shared information. 2018 21st International Conference on Intelligent Transportation Systems (ITSC) pp. 3961–3966 (2018), `https://api.semanticscholar.org/CorpusID:54460348`
25. Rockl, M., Strang, T., Kranz, M.: V2v communications in automotive multi-sensor multi-target tracking. In: 2008 IEEE 68th Vehicular Technology Conference. pp. 1–5. IEEE (2008)
26. Rong, Y., Bian, Y., Xu, T., yang Xie, W., Wei, Y., bing Huang, W., Huang, J.: Self-supervised graph transformer on large-scale molecular data. arXiv: Biomolecules (2020), `https://api.semanticscholar.org/CorpusID:226191736`
27. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
28. Wang, T., Chen, G., Chen, K., Liu, Z., Zhang, B., Knoll, A., Jiang, C.: Umc: A unified bandwidth-efficient and multi-resolution based collaborative perception framework. arXiv preprint arXiv:2303.12400 (2023)
29. Wang, T.H., Manivasagam, S., Liang, M., Yang, B., Zeng, W., Urtasun, R.: V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. In: Computer

Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16. pp. 605–621. Springer (2020)

30. Wu, Z., Jain, P., Wright, M.A., Mirhoseini, A., Gonzalez, J., Stoica, I.: Representing long-range context for graph neural networks with global attention. ArXiv **abs/2201.08821** (2022), https://api.semanticscholar.org/CorpusID:246210055
31. Xiang, H., Xu, R., Ma, J.: Hm-vit: Hetero-modal vehicle-to-vehicle cooperative perception with vision transformer. arXiv preprint arXiv:2304.10628 (2023)
32. Xu, R., Chen, W., Xiang, H., Xia, X., Liu, L., Ma, J.: Model-agnostic multi-agent perception framework. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 1471–1478. IEEE (2023)
33. Xu, R., Guo, Y., Han, X., Xia, X., Xiang, H., Ma, J.: Opencda: An open cooperative driving automation framework integrated with co-simulation. 2021 IEEE International Intelligent Transportation Systems Conference (ITSC) pp. 1155–1162 (2021), https://api.semanticscholar.org/CorpusID:260953729
34. Xu, R., Li, J., Dong, X., Yu, H., Ma, J.: Bridging the domain gap for multi-agent perception. In: 2023 IEEE International Conference on Robotics and Automation (ICRA). pp. 6035–6042. IEEE (2023)
35. Xu, R., Tu, Z., Xiang, H., Shao, W., Zhou, B., Ma, J.: Cobevt: Cooperative bird's eye view semantic segmentation with sparse transformers. arXiv preprint arXiv:2207.02202 (2022)
36. Xu, R., Xia, X., Li, J., Li, H., Zhang, S., Tu, Z., Meng, Z., Xiang, H., Dong, X., Song, R., et al.: V2v4real: A real-world large-scale dataset for vehicle-to-vehicle cooperative perception. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13712–13722 (2023)
37. Xu, R., Xiang, H., Tu, Z., Xia, X., Yang, M.H., Ma, J.: V2x-vit: Vehicle-to-everything cooperative perception with vision transformer. In: European conference on computer vision. pp. 107–124. Springer (2022)
38. Xu, R., Xiang, H., Xia, X., Han, X., Li, J., Ma, J.: Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 2583–2589. IEEE (2022)
39. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10), 3337 (2018)
40. Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Lin, H., Zhang, Z., Sun, Y., He, T., Mueller, J., Manmatha, R., et al.: Resnest: Split-attention networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2736–2746 (2022)
41. Zhang, H., Luo, G., Li, Y., Wang, F.Y.: Parallel vision for intelligent transportation systems in metaverse: Challenges, solutions, and potential applications. IEEE Transactions on Systems, Man, and Cybernetics: Systems **53**(6), 3400–3413 (2023). https://doi.org/10.1109/TSMC.2022.3228314
42. Zhang, J., Zhang, H., Sun, L., Xia, C.: Graph-bert: Only attention is needed for learning graph representations. ArXiv **abs/2001.05140** (2020), https://api.semanticscholar.org/CorpusID:210698881
43. Zhang, X., Zhang, A., Sun, J., Zhu, X., Guo, Y.E., Qian, F., Mao, Z.M.: Emp: edge-assisted multi-vehicle perception. Proceedings of the 27th Annual International Conference on Mobile Computing and Networking (2021), https://api.semanticscholar.org/CorpusID:238997956

44. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4490–4499 (2018)

6 Appendix

6.1 Calculation of the Spatial Range of Pillar Regions Corresponding to Feature Points

As stated in Sec. 3.2, the spatial correlation between nodes needs to be judged according to whether the bottom surface of the cylindrical space area corresponding to the nodes is intersected. We calculate the spatial range of the bottom surface of the pillar regions $\mathcal{M}(\mathcal{P})$ corresponding to the feature points based on the voxel grid size settings in the model parameters and the ratio of the BEV feature dimensions before and after refinement by the backbone network. Below is the detailed process:

If the voxel grid size in the model parameters is set to $[v_x, v_y, v_z]$, the size of the initial BEV feature is set to $\mathcal{F}_{init}$ is $[C_1, H_1, W_1]$, and the size of the BEV feature $\mathcal{F}_{refine}$ after refined by the backbone network is set to $[C_2, H_2, W_2]$. Thus, for the feature point at index (i, j) within $\mathcal{F}_{refine}$, the size of the bottom surface of the corresponding cylindrical space $\mathcal{M}(\mathcal{P})$ is determined by:

$$len_x = v_x \cdot \frac{W_1}{W_2} \tag{16}$$

$$len_y = v_y \cdot \frac{H_1}{H_2} \tag{17}$$

the coordinates (x, y) representing the center point of the bottom surface of the cylindrical space region in the Cartesian coordinate system is determined by:

$$x = \left(j + 0.5 - \frac{W_2}{2}\right) \cdot len_x \tag{18}$$

$$y = \left(\frac{H_2}{2} - i - 0.5\right) \cdot len_y \tag{19}$$

where Cartesian coordinate system takes the intelligent vehicle ego as the coordinate origin, and the front and right sides of the ego are the positive directions of the Y-axis and the X-axis, respectively.

Given the size (len_x, len_y) and the central coordinates (x, y) of the cylindrical space region $\mathcal{M}(\mathcal{P})$, we can determine whether there is an intersection between the bottom surfaces of the pillar regions corresponding to the feature points, which reveals the spatial relationships between the feature points, thereby enabling the construction of the feature collaboration graph.

It is worth noting that, due to the uncertainty in model parameter settings and BEV feature sizes, we have also developed a dedicated CUDA operator to calculate the range of the bottom surface of the spatial regions $\mathcal{M}(\mathcal{P})$ corresponding to feature points in real-time.

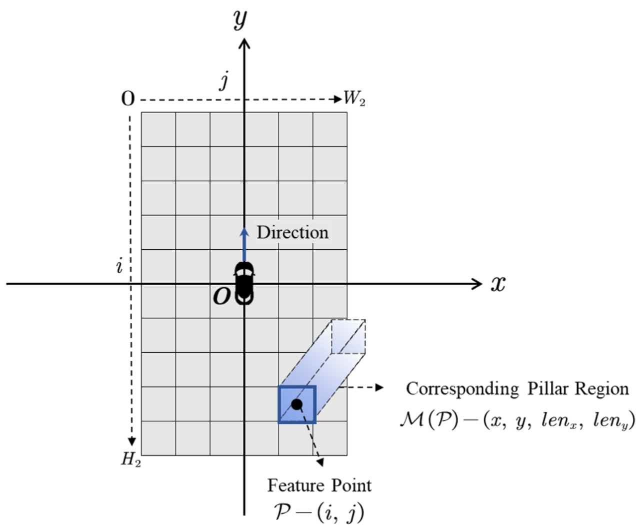

y
x
j
0
W_2
i
H_2
O
Direction
Corresponding Pillar Region
$\mathcal{M}(\mathcal{P})-(x,\ y,\ len_x,\ len_y)$
Feature Point
$\mathcal{P}-(i,\ j)$

Fig. 7: Visualization of spatial regions corresponding to feature points. The gray grid represents the the bottom surface of the pillar region $\mathcal{M}(\mathcal{P})$ corresponding to the feature points, and the blue cylinders represent the cylindrical spatial regions corresponding to the feature points $\mathcal{P}$.

6.2 Semantic Mapper

In the feature collaboration graph, the number of channels and the semantic space exists difference between the nodes $\left\{\mathcal{P}_n^j\right\} \subset \mathbb{R}^{C_j}$ from neb and the nodes $\left\{\mathcal{P}_m^i\right\} \subset \mathbb{R}^{C_i}$ from ego. To address this, we have designed a semantic mapper, inspired by the autoencoder [2] structure. This mapper adjusts the number of channels and semantic space of neb nodes to align with the architecture of ego nodes, as depicted in Fig. 8.

Network Structure. The semantic mapper is made up of two sets of 1x1 convolutions. For a node $\mathcal{P}_n^j$ from neb, its dimension is initially reduced by the first group of two 1x1 convolutions, resulting in a low-dimensional hidden variable. Then, this hidden variable's dimension is increased to match that of the ego node $\mathcal{P}_m^i$ using another set of two 1x1 convolutions in the second group. In this way, the number of node channels and the semantic space in the feature collaboration graph are unified.

There are various types of architectures for the observation encoder used by agents (such as PointPillar [10], SECOND [39], etc.). We train a standard semantic mapper $\mathcal{H}_{map_{ego}}^{(arc)}(\cdot)$ for each type of observation encoder architecture.

Channel Aligner. In real scenarios, the intermediate features extracted by observation encoders with the same architecture may still have slight differences

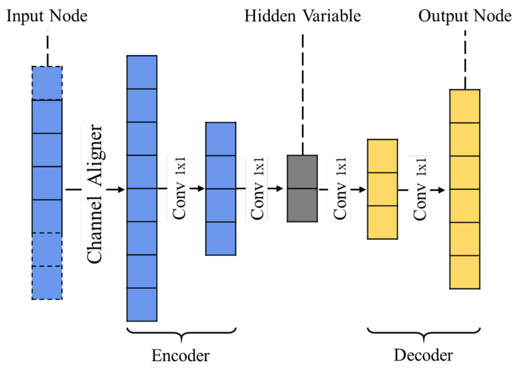


Fig. 8: Semantic mapper. The two sets of 1x1 convolutions is used to map the node representation to be the same as that of the ego node. Before the convolution layers, a Channel Aligner adapts to slight variations in the number of channels by randomly dropping or padding channels.

in the number of channels due to different parameter settings. To tackle this problem, we integrate a channel aligner [34] before the $\mathcal{H}^{(arc)}_{map_{ego}}(\cdot)$ to adapt to minor changes in the number of feature channels.

For node $\mathcal{P}^j_n$ from neb, if the number of channels matches the input dimension of the standard semantic mapper $\mathcal{H}^{(arc)}_{map_{ego}}(\cdot)$, it is directly inputted into the mapper. However, if the number of channels does not align with the input dimension of $\mathcal{H}^{(arc)}_{map_{ego}}(\cdot)$, channels are either added or removed at random to ensure the channel number of $\mathcal{P}^j_n$ matches the standard mapper's input dimension, and input to $\mathcal{H}^{(arc)}_{map_{ego}}(\cdot)$. The integration of a channel aligner enhances Hetecooper's adaptability in real-world applications, allowing it to automatically adjust to minor variations in feature channel numbers without the need to retraining the model.

Training. In Hetecooper's collaboration framework, the semantic mapper and the fusion module's parameters are adjusted during the second phase of training, as described in Sec. 3.4.

6.3 Collaboration Methods of Baseline Under Model Heterogeneous Conditions.

In Baseline's Where2comm [8] and Fcooper [4], the collaborative method is designed based on the assumption of model homogeneity, which doesn't support direct fusion of heterogeneous features. To adapt the baseline method for models heterogeneous conditions, we first standardize the feature channel numbers and semantic spaces using the semantic mapper mentioned in Sec. 6.2 in Appendix. Subsequently, we align the feature dimensions using bilinear interpolation methods. After unifying the feature representations, we then use the original fusion module to merge the features from collaborators.

At this point, the training process of the baseline also consists of two stages: First, we train the observation encoder, fusion module, and detection head under the setting of model homogeneity. Second, we adapt a $\mathcal{H}_{map_{ego}}^{(arc)}(\cdot)$ for each observation encoder architecture. Here's the detail process:

i) Input the perception data into the observation encoders $\mathcal{H}_{enc_{ego}}(\cdot)$ for the ego and $\mathcal{H}_{enc_{neb}}(\cdot)$ for the neb, respectively. Then the intermediate features $\mathcal{F}_{ego}$ and $\mathcal{F}_{neb}$ is yield.

ii) Input the neb feature $\mathcal{F}_{neb}$ into the semantic mapping module $\mathcal{H}_{map_{ego}}^{(arc)}(\cdot)$ to align the number of channels and the semantic representation, resulting in $\mathcal{F}'_{neb} = \mathcal{H}_{map_{ego}}^{(arc)}(\mathcal{F}_{neb})$.

iii) Adjust the size of $\mathcal{F}'_{neb}$ to match $\mathcal{F}_{ego}$ through bilinear interpolation, resulting in $\mathcal{F}''_{neb} = \text{Bilinear}(\mathcal{F}'_{neb})$.

iv) Calculate the root mean square error between $\mathcal{F}_{ego}$ and $\mathcal{F}''_{neb}$ to serve as the training loss:

$$L = MSE\left(\mathcal{F}_{ego}, \mathcal{F}''_{neb}\right)$$

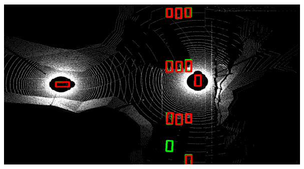
v) Update the parameters of the feature mapper $\mathcal{H}_{map_{ego}}^{(arc)}(\cdot)$, while keeping the parameters of observation encoders $\mathcal{H}_{enc_{ego}}(\cdot)$ and $\mathcal{H}_{enc_{neb}}(\cdot)$ unchanged.

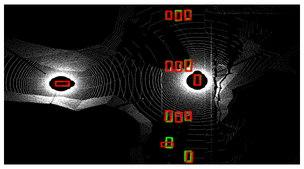
6.4 Multi-scale Collaboration.

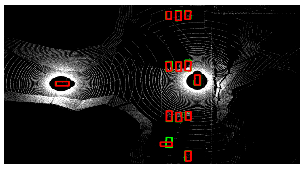
There exist voxel-level and object contour-level correlations between features. To capture richer correlations between heterogeneous features, we construct feature collaboration graphs at multiple spatial scales. The detailed process is as follows:

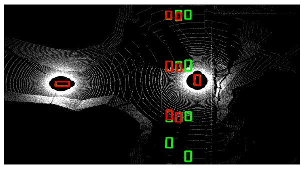
For the feature $\mathcal{F}_i \in \mathbb{R}^{H_i \times W_i \times C_i}$ from ego and the information $\mathcal{I}_{j \to i} \in \mathbb{R}^{H_j \times W_j \times C_j}$ from neb. We divide them into tensors with dimensions of $(C_i \times P_s \times P_s, \frac{H_i}{P_s}, \frac{W_i}{P_s})$ for ego and $\left(C_j \times P_s \times P_s, \frac{H_j}{P_s}, \frac{W_j}{P_s}\right)$ for neb, using the patch size $P_s \times P_s$, where $s = 1, 2, ..., S$. These tensors can be thought as hyperfeatures with size $\left(\frac{H_i}{P_s}, \frac{W_i}{P_s}\right)$ and $\left(\frac{H_j}{P_s}, \frac{W_j}{P_s}\right)$, respectively. Each feature points in the hyperfeatures are patches of size $P_s \times P_s$, which we called as hyperpoints. The spatial region corresponding to the hyperpoints is the sum of the cylindrical grid regions corresponding to all the feature points in the patch $P_s \times P_s$.

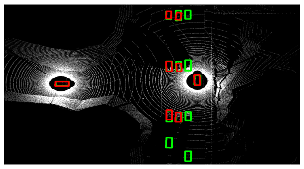
Further, we follow the method in Sec. 3.2 to construct a feature collaboration graph between hyperfeatures across various scales and then propagate and aggregated feature messages. And then the fusion result $\mathcal{F}_{ij}^s$ is obtained. During this stage, the attention mechanisms is executed separately between the feature points at the corresponding position of the patch $P_s \times P_s$. Ultimately, we employ the Split-atten method [40] to combine the results $\mathcal{F}_{ij}^s$ at various scales, producing $\mathcal{F}'_{ij} = \text{Split}\left(\left\{\mathcal{F}_{ij}^s\right\}_{s=1}^{S}\right)$. This process effectively aggregates the information of collaborator features at multiple scales into ego features.


(a) Hetecooper


(b) MDPA


(c) HM-ViT


(d) No Fusion


(e) Where2comm

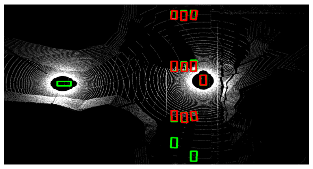
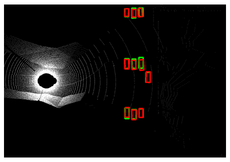
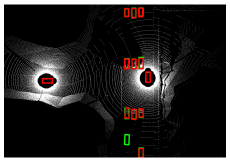
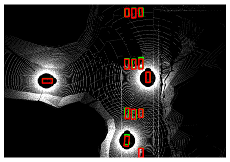

(f) Fcooper

Fig. 9: Visual comparison of detection results of different collaborative methods. The green and red boxes represent the ground-truth and the detection results by different collaboration methods, respectively.


(a) 1 Vehicle


(b) 2 Vehicles


(c) 3 Vehicles

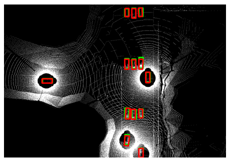

(d) 4 Vehicles

Fig. 10: Visual comparison of Hetecooper detection results when the number of intelligent vehicles increases from 1 to 4. The green and red boxes represent the ground-truth and the detection results by different collaboration methods, respectively.

6.5 Visualization of Detection Results

We visualized the detection results after collaborating using different methods. The ego (vehicle in left-hand) and the neb (vehicle in right-hand) employs PointPillar and SECOND observation encoder architecture, respectively.

Fig. 9a-h shows the visualizations of detection results witour collaboration and with collaboration using Hetecooper, HM-ViT [31], MDPA [34], Where2comm [8], Fcooper [4], respectively. It is easy to see that Hetecooper achieved the best performance.

Furthermore, we evaluated the detection performance of Hetecooper as the number of intelligent vehicles in the scene increased from 1 to 4, as shown in Fig. 10a-d.

6.6 Test Results on the Real Dataset V2V4Real

V2V4Real [36] is the first large-scale real-world dataset for Vehicle-to-Vehicle (V2V) cooperative perception in autonomous driving. It is collected by two vehicles simultaneously in the same location, and contains 410 km of the driving area, 20K LiDAR, 40K RGB, and 240K annotated 3D bounding boxes across 5 vehicle classes. Intersections, highway entrance ramps, and straight city roads are concluded.

We tested the performance of Hetecooper and the baselines on V2V4Real when agents used heterogeneous observation encoders, as shown in Tab. 5.

Table 5: Comparison when architecture of observation encoders are different on V2V4Real. The terms on the left and right sides of the "-" represent the names of the observation encoders used for the ego and neb, respectively. Results are reported in AP@0.50/AP@0.70. Since the performance of the collaboration method is related to the proportion of heterogeneous agents in the scenario, we uniformly set the maximum number of vehicles in the scenario to 2 to eliminate the influence of unrelated variables.

Observation Encoder	Collaborative Method				
	Hetecooper	HM-ViT	MDPA	Where2comm	Fcooper
SECOND	**0.601**/**0.508**	0.579/0.364	0.594/0.482	0.594/0.482	0.5800/0.441
PointPillar	0.607/**0.465**	0.607/0.400	**0.612**/0.426	0.612/0.426	0.608/0.408
PointPillar - SECOND	**0.569**/**0.443**	0.543/0.425	0.543/0.423	0.550/0.428	0.558/0.415