CoPe: Taming Collaborative 3D Perception via Lite Network Attention across Mobile Agents

Shifan Zhang*, Hongzi Zhu*✉, Yunzhe Li*, Liang Zhang†, Shan Chang†, Minyi Guo*
* *Shanghai Jiao Tong University, Shanghai, China*
† *Donghua University, Shanghai, China*
{zhangshifan, hongzi, yunzhe.li}@sjtu.edu.cn, {zhangliang, changshan}@dhu.edu.cn, guo-my@cs.sjtu.edu.cn

***Abstract*—To extend the receptive field of a mobile agent in complex scenarios, it is essential for multiple agents to cooperate with each other. However, it is challenging to achieve comprehensive 3D perception at the minimal computational and communication costs. In this paper, we propose *CoPe*, a lightweight and efficient collaborative 3D perception scheme for mobile agents. The main idea of CoPe is for an ego agent to query the *most helpful* information from its neighboring agents through a lightweight network attention mechanism. To this end, at each agent, we first leverage Singular Value Decomposition (SVD) to decompose a full-size point cloud feature into components. Meanwhile, with the novel self-attention and cross-attention algorithms, we respectively select the key component of an ego agent that best represent the point cloud of the ego agent as a query, and valuable components of each helper agent that are most relevant to the query as the answer. After feature reconstruction and aggregation, an ego agent can have a comprehensive understanding about the scene and make accurate predictions on downstream tasks. CoPe is lightweight and can be easily implemented on mobile devices. Results of extensive experiments conducted on both real-world and simulation datasets demonstrate that CoPe can achieve superior 3D object detection accuracy while significantly reducing the incurred computational and communication costs.**

***Index Terms*—collaborative perception, network attention, singular value decomposition**

I. Introduction

It is essential for mobile agents, such as unmanned aerial vehicles (UAVs), robots, and autonomous vehicles, to obtain comprehensive and accurate three-dimension (3D) perception of the environment [1]–[3]. As Light Detection and Ranging (LiDAR) sensors become much smaller and significantly less costly, it is prevalent to deploy one or more LiDARs on a mobile agent to constantly scan the area around the agent, obtaining reflected signals, referred to as *point clouds* about the environment for precise 3D perception. It is inevitable, however, that objects of interest in a scene may occlude each other from the perspective of a mobile agent, causing false negative detection errors and serious consequences, especially when the agent and/or objects are moving fast. For example, in the U.S., more than $840,000$ traffic accidents are due to blind spots of vehicles each year [4]. To achieve comprehensive 3D scene perception, collaborative perception among mobile agents is the key, where adjacent agents can exchange necessary information via wireless communication, substantially extending the perceptual range and precision of each agent in the end.

A practical collaborative 3D perception algorithm in mobile settings needs to fulfill the following three stringent objectives. First, the algorithm should attain a *high 3D perception accuracy* for each participating agent. This pertains even in scenarios where there exist objects, differing in quantity and type, that are positioned at disparate distances and obstructed to different extents. Second, the algorithm should be *lightweight* in terms of both computational and communication overheads. As battery-powered mobile agents have constrained computing power and limited bandwidth on wireless links, intensive model inference and exchanging high-volume data are prohibited. Last but not least, the algorithm should be agile with a *low response time*. Otherwise, results may be out of date for making online decisions on moving agents.

In the literature, existing pioneer algorithms can be classified into three categories, *i.e.*, early fusion based, intermediate fusion based, and late fusion based. Early fusion based algorithms [5] [6] directly exchange and align raw point cloud data among agents through point cloud registration (PCR) or other given localization methods. Then, the fused point cloud is used to extract features for downstream tasks. These methods can obtain superior 3D perception accuracy but incur enormous computational and communication overheads. In contrast, late fusion based algorithms [7] exchange and fuse individual 3D perception results, such as 3D bounding boxes of identified objects, by conducting efficient graph matching on constructed multi-affinity graphs. Though such algorithms have low communication cost, the improvement in perception accuracy is rather limited due to the lack of semantic information about the scene [8], [9] and inaccurate individual perception results. Recently, intermediate fusion based algorithms have received great attention, where intermediate features are selected based on identified objects [10] or compressed [8], [11], [12] before being shared to peer agents. Similarly, the inaccuracy of individual detection results and the information loss caused by compression limit the perception accuracy of these algorithms.

In this paper, we introduce *CoPe*, a novel *Co*llaborative 3D *Pe*rception algorithm for mobile agents. The core idea of CoPe is for an ego agent to leverage a lightweight network attention mechanism to obtain the *most helpful* information from its neighboring agents, so that comprehensive 3D perception can be achieved at the minimal computational and communication

✉ Corresponding author

costs. More specifically, an ego agent first establish a *hint* of compact size that can most represent the current observed scene, and broadcasts it out to query neighboring agents. Upon receiving a query, those helping agents select the most related and helpful feature from their own perspectives based on the attention calculated on the received query and features extracted from their local point clouds.

Two main challenges are encountered in the design of CoPe. First, it is challenging to design an effective query that can best represent the current scene (*i.e.*, the newly collected point cloud) of an ego agent and has the smallest size for transmission at the same time. To tackle this challenge, the ego agent first extracts a full-size feature from the point cloud using a pre-trained backbone network, and decomposes the feature into a series of components through Singular Value Decomposition (SVD). We have an insight that *components with largest singular values do not necessarily contribute the most* to a specific downstream perception task. Therefore, instead of selecting the top-k components according to singular value ranking as traditional data compression and reconstruction do, the ego agent employs a multi-head self-attention operation to assess the significance of each SVD component with respect to the downstream task. The component with the highest self-attention score is selected as the scene hint to query other agents.

Second, it is non-trivial for helper agents to determine what information they have can improve the 3D perception accuracy of a querying agent. To deal with this challenge, one or multiple helper agents also conduct the same SVD on their point cloud features and try to find key components that are most correlated with the received query. Specifically, each helper agent first uses the received query to conduct a multi-head cross-attention operation over its own SVD components, and selects components that meet a relevance threshold. Moreover, to supplement the observation loss of the ego agent due to the blockage of the line of sight, each helper agent also conducts the same multi-head self-attention operation as introduced above to select key components that this helper agent regards as important. Finally, given the real-time bandwidth constraint, key components with largest weights are sent back to the querying agent for feature fusion and downstream tasks.

We implement CoPe on three mobile platforms with varying computational power. We consider 3D object detection task on one real-world dataset and two simulation datasets involving a rich set of scenarios with diverse environment. CoPe is lightweight and agile which consumes about 49MB memory and has a low mean response time of 60ms even running on a Nvidia Jetson TX2 NX. Extensive experiments have been conducted and the results show that, on the DAIR-V2X [13], OPV2V [14] and CoPeSet datasets, CoPe outperforms SCOPE, a state-of-the-art method [12], by 12.94%, 9.93% and 13.04% in terms of detection precision, while reducing the communication volume by 88.78%, 89.52% and 89.28%, respectively.

We highlight the main contributions made in this work as follows:

- We have proposed a novel lightweight network attention mechanism, utlizing SVD components as query and keys, to improve both the efficiency and accuracy of collaborative 3D perception for mobile agents.
- We have implemented CoPe on different mobile devices and conducted extensive trace-driven simulations, demonstrating the efficacy of CoPe design.

II. Problem Definition

A. System Model

A mobile agent can be in one or two roles, *i.e.*, *ego* and *helper*, at the same time. An ego agent needs to gain comprehensive 3D scene information based on its own sensor readings or the querying results sent from one or multiple neighboring helper agents. A helper agent acts upon the request of an ego agent and returns appropriate information back to the ego agent. In addition to mobile agents, a helper can also be a roadside unit equipped with sensing and communication capabilities. Moreover, we consider a physical mobile agent have the following capabilities:

- **Sensing:** Agents are equipped with LiDAR sensors and can obtain real-time point cloud data about the scene. In addition, agents can also acquire high-precision pose and location information in the global coordination system via other localization schemes.
- **Computation:** Agents have sufficient computing power to deal with normal matrix operations and model inference of deep neural network (DNN) of moderate size.
- **Communication:** Agents can exchange information via short-range wireless communication such as C-V2X PC5 interface [15]. The bandwidth of a wireless link can vary over time and the density of agents in vicinity may be high.

B. Collaborative 3D Perception Problem

We focus on collaborative 3D perception in terms of 3D object detection using point cloud data and define the problem as

$$\begin{aligned} \min_{\theta,\mathbf{F},\{\mathbf{F}_i\}} \quad & \ell(\hat{\mathbf{Y}},\mathbf{Y}), \\ \text{s.t.} \quad & \hat{\mathbf{Y}} = \mathcal{F}_\theta(\mathbf{F},\{\mathbf{F}_i|i=1,2,...,n\}), \\ & \sum_{i=1}^{N}\mathcal{G}(\mathbf{F}_i) \le B, \end{aligned} \tag{1}$$

where $\ell(\cdot)$ represents the detection error function; $\hat{\mathbf{Y}}$ denotes the result of all predicted objects in a scene of an ego agent, and $\mathbf{Y}$ is the corresponding ground truth; $\mathcal{F}_\theta(\cdot)$ denotes the DNN detection model parameterized by θ; $\mathbf{F}$ denotes the feature extracted from the local point cloud of the ego agent and $\mathbf{F}_i$ for $i=1,2,...,n$ denotes the reconstructed feature of each of the n helper agents, respectively; $\mathcal{G}(\cdot)$ measures the communication cost and B is the available bandwidth of a wireless channel.

III. Empirical Study

In this section, we investigate the information loss of using compressed feature in representing a point cloud scene with respect to the 3D object detection task. To control the impact of the compressed ratio to the performance of detection task, we apply Singular Value Decomposition (SVD) to the feature obtained from a point cloud feature extraction backbone, and study the utility of each derived component. If a small number of key components can achieve satisfactory detection accuracy, it is feasible to use these key components to represent a full-size feature.

A. Preliminaries of Singular Value Decomposition

Singular Value Decomposition (SVD) is a powerful mathematical technique that decomposes a matrix into three distinct matrices, revealing the intrinsic structure and key properties of the original matrix. Formally, for a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ where $m < n$ without loss of generality, the SVD of the matrix $\mathbf{A}$ can be expressed as: $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}$, where $\mathbf{U} \in \mathbb{R}^{m \times m}$ and $\mathbf{V} \in \mathbb{R}^{n \times n}$ are two orthonormal matrices, representing the left and right singular vectors, respectively. $\mathbf{\Sigma} \in \mathbb{R}^{m \times n}$ is a diagonal matrix that contains the singular values of $\mathbf{A}$, sorted in descending order. Further, let $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m]$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n]$, the matrix $\mathbf{A}$ can be reformulated as:

$$\mathbf{A} = \sum_{i=1}^{m} \sigma_i \mathbf{u}_i \mathbf{v}_i^\mathsf{T}, \tag{2}$$

where $\sigma_i = \mathbf{\Sigma}_{ii}$ denotes the i-th singular value of the matrix $\mathbf{A}$. Therefore, each term $\sigma_i \mathbf{u}_i \mathbf{v}_i^\mathsf{T}$ is referred to as a *component* of the matrix $\mathbf{A}$.

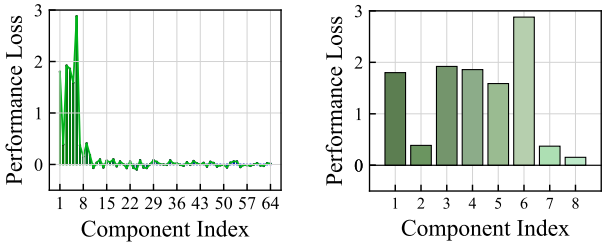
In SVD, since the largest singular values correspond to the components that encapsulate the majority of the informational content in a matrix, while each subsequent singular value contributes progressively less information [16], this enables efficient dimensionality reduction and noise filtering. By retaining only the most significant components of a matrix, we can effectively preserve its core structure and essential information while discarding redundant or less meaningful content. Therefore, the matrix can be approximated by retaining the top k components with the largest singular values as follows:

$$\mathbf{A} \approx \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^\mathsf{T}. \tag{3}$$

B. Utility of Each SVD Component

As we have discussed, the components associated with larger singular values encapsulate the most critical information in a matrix, enabling effective reconstruction of the matrix. In contrast, in the context of 3D perception tasks, the most important components are those that enable the model to detect more objects and improve detection accuracy. We conduct intensive analysis to answer the question: *are the components corresponding to larger singular values inherently more important for 3D perception tasks?*

To answer this question, we conduct 3D object detection on the test set of the OPV2V [14] dataset, which serves as an open benchmark for collaborative perception tasks under vehicle-to-vehicle (V2V) communication scenarios. The dataset is constructed using simulations from OpenCDA [17] and CARLA [18]. The test set comprises $2,719$ frames of annotated 3D point clouds. The number of vehicles per frame ranges from 2 to 7. The PointPillars framework [1] is adopted on all point clouds collected on each vehicle in each frame. More specifically, we first apply singular value decomposition (SVD) on each feature extracted with the feature extraction backbone employed in the PointPillars framework. Then, we rank the components on the basis of their singular values. For each time, we remove one component from the component set and reconstruct the feature and conduct the downstream object detection task. We calculate the performance loss due to the omission of a certain component by subtracting the detection accuracy obtained with the reconstructed feature from that obtained with the original feature.



(a) All Components (b) First 8 Components

Fig. 1: 3D object detection accuracy loss analysis of omitting each SVD component when reconstructing a feature using SVD components. Components are ranked in descending order according to their corresponding singular values. (a) shows the performance loss across all 64 components; (b) zooms in on the first 8 components from (a) for clearer visualization.

Fig. 1 depicts the average performance loss as a function of omitting components across all frames in the OPV2V test set. Fig. 1(a) presents the performance loss over all 64 components. We observe that removing components with indices larger than 8 leads to negligible changes in object detection performance, indicating that components corresponding to extremely smaller singular values contain minimal useful information for the task. To enhance readability and facilitate focused analysis, Fig. 1(b) illustrates a subset of the results, highlighting only the first 8 components shown in (a).

Moreover, based on the performance loss observed when retaining only the first 8 components, we derive the following insight:

Observation. *Components with the large singular values are not necessarily the most critical components for the downstream task.*

Therefore, it is not straightforward to select the optimal components to represent a point cloud with respect to a particular perception task.

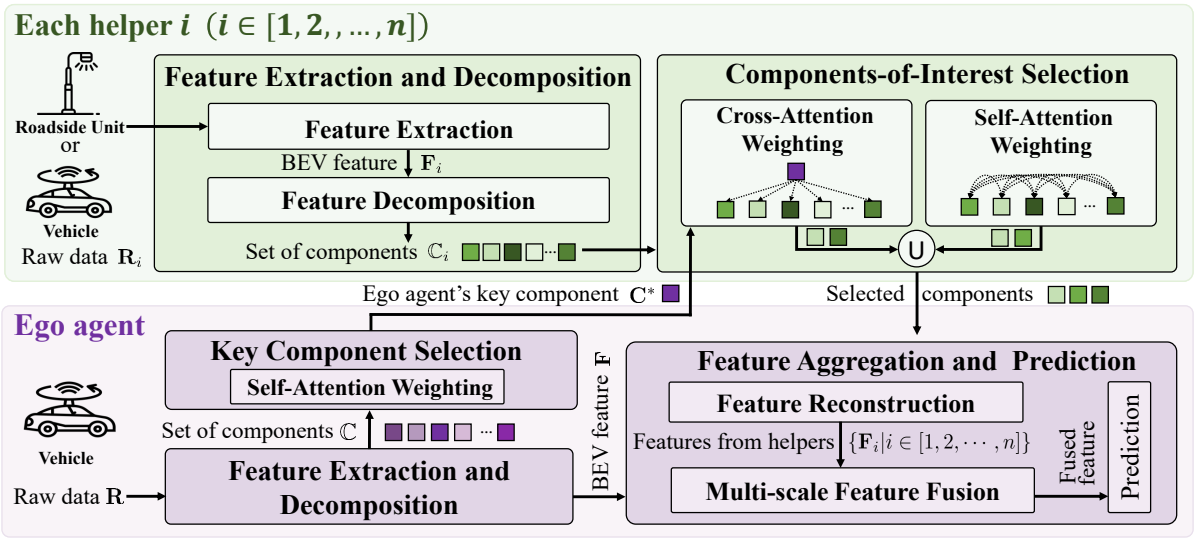


Fig. 2: Architecture of CoPe, consisting of two agent roles. The ego agent, typically a mobile agent such as a vehicle, mainly include three modules, *i.e.*, feature extraction and decomposition (FED), key component selection (KCS) and feature aggregation and prediction (FAP). FED is designed for feature extraction and conducting SVD decomposition of the feature; then KCS selects the most representative component as a query to request helper agents; finally, FAP reconstructs and fuses all received useful information and make prediction. The helper agent, which can be either another mobile agent or a roadside unit, shares the same FED module and additionally includes a components-of-interest selection (CIS) module. CIS identifies most useful components based on the query of the ego agent and the self-assessment of the helper agent.

IV. Design of CoPe

A. Overview

The core idea of CoPe is for an ego agent to leverage a lightweight network attention module to obtain the *most helpful* information from its neighboring agents, so that comprehensive 3D perception can be achieved at the minimal computational and communication costs. The overall architecture of CoPe is illustrated in Fig. 2, which consists of three modules in ego agent and two modules in helper agent roles.

An ego agent, typically a mobile agent, integrates three modules, *i.e.*, Feature Extraction and Decomposition (FED), Key Component Selection (KCS), and Feature Aggregation and Prediction (FAP).

Feature Extraction and Decomposition (FED): In this module, the raw point cloud data $\mathbf{R}$ collected by an ego agent are first processed into Bird's Eye View (BEV) features, denoted as $\mathbf{F}$. These features are subsequently partitioned into multiple components through Singular Value Decomposition (SVD), which form the set $\mathbb{C}$. Each component $\mathbf{C} \in \mathbb{C}$ has a significantly smaller communication volume, ensuring low communication overheads.

Key Component Selection (KCS): To design an effective query that can best represent the current scene of the ego agent, a key component $\mathbf{C}^* \in \mathbb{C}$ is selected through Self-Attention Weighting (SAW) module. Specifically, a multi-head self-attention operation is employed on the ego agent's component set $\mathbb{C}$. Then the component with the highest self-attention score is selected as the scene hint. The hint encapsulates the ego agent's requirements and is then broadcast to its helper agents as a query for further processing.

Feature Aggregation and Prediction (FAP): Upon receiving components from each helper agent, the ego agent first reconstructs the features selected by the helper agents and then integrates these features with its own features to derive a fused feature. To ensure effective integration of both low-level and high-level features, thereby improving the overall fusion quality, a multi-scale fusion approach is employed. This fused feature is then utilized to generate the final 3D detection results.

A helper agent, which can be either another mobile agent or a roadside unit, contains two modules, *i.e.*, Feature Extraction and Decomposition (FED) and Components-of-Interest Selection (CIS). The FED module is the same as the one introduced in ego agent and it derives the component set based on the raw point cloud data observed by helper agents.

Components-of-Interest Selection (CIS): The primary purpose of the CIS module is to identify the most critical components of the helper agent that can enhance the ego agent's 3D perception accuracy, and transmit them to the ego agent. It encompasses two aspects: first, each helper agent selects relevant components in response to the query $\mathbf{C}^*$ from the ego agent. Second, to compensate for the ego agent's observation loss due to line-of-sight obstructions, each helper agent also selects components that it deems important.

To achieve these, we propose the Cross-Attention Weighting (CAW) module, as well as Self-Attention Weighting (SAW) module. Specifically, consider the i-th helper agent as an

example: The CAW module performs a multi-head cross-attention operation between the ego agent's query $\mathbf{C}^*$ and i-th helper agent's component set $\mathbb{C}_i$, generating a cross-attention score. A relevance threshold is then applied to select the components most relevant to the ego agent's query. Meanwhile, the SAW module conducts a multi-head self-attention operation on $\mathbb{C}_i$ to derive a self-attention score, with a significance threshold used to select the components that the i-th helper agent deems most important. Finally, given the real-time bandwidth constraint, components with largest scores are sent back to the ego agent for feature fusion and downstream tasks.

B. Ego Agent

As previously mentioned, the ego agent side primarily consists of three modules: Feature Extraction and Decomposition (FED), Key Component Selection (KCS), and Feature Aggregation and Prediction (FAP).

1) Feature Extraction and Decomposition: Consistent with prior research [10], [19], feature extraction backbone employed in PointPillars [1] is adopted to convert the raw point cloud data $\mathbf{R}$ collected by the ego agent into BEV features $\mathbf{F} \in \mathbb{R}^{C\times H\times W}$. This transformation simplifies the representation of spatial information, making it more efficient and suitable for fusion processes.

Subsequently, the feature tensor $\mathbf{F}$ is reshaped into a matrix of dimensions $C \times HW$. SVD is then applied to decompose the matrix $\mathbf{F}$, resulting in a set $\mathbb{C}$ containing the top k components:

$$\mathbb{C} = \{\sigma_m \mathbf{u}_m \mathbf{v}_m^{\mathsf{T}} | m = 1, 2, ..., k\}. \tag{4}$$

2) Key Component Selection: To design an effective query that best represents the current scene of the ego agent, a key component $\mathbf{C}^* \in \mathbb{C}$ is selected through the Self-Attention Weighting (SAW) module.

The SAW module primarily consists of a multi-head self-attention operation applied to the ego agent's component set, which is then used to derive a self-attention score for each component. The component with the highest self-attention score is selected as the scene hint, effectively capturing the most relevant information for the ego agent's current perception of the scene.

Specifically, first, the obtained k components, each with dimensions $C\times HW$, are processed using Global Max Pooling along the HW dimension. In other words, for each $C \times HW$ component matrix, the maximum value is computed for each row, resulting in a C-dimensional vector. This C-dimensional vector encapsulates the global information of the corresponding component [20].

Next, the $k \times C$ matrix $\mathbf{Q}$, representing the components, are fed into Self-Attention Weighting (SAW) module. This SAW module is designed based on Multi-Head Self-Attention (MHSA) [21] to learn the interrelationships and relative significance of the components, effectively capturing the dependencies and importance of each component with respect to the downstream task. Specifically, the computation of SAW is defined as follows:

$$\begin{aligned} \mathrm{SAW}(\mathbf{Q}) &= \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)\mathbf{W}^O, \\ \mathrm{head}_s &= \mathrm{Attn}(\mathbf{W}_s^Q\mathbf{Q}, \mathbf{W}_s^K\mathbf{Q}) \\ &= \mathrm{softmax}\left(\frac{\mathbf{W}_s^Q\mathbf{Q}(\mathbf{W}_s^K\mathbf{Q})^{\mathsf{T}}}{\sqrt{C}}\right), \end{aligned} \tag{5}$$

where $\mathbf{W}^O$, $\mathbf{W}_s^Q$ and $\mathbf{W}_s^K$ for $s = 1, ...h$ are trainable parameters. h is the number of heads. The final output of the SAW assigns a self-attention score to each component. Since the entire SAW module is trained end-to-end, these self-attention scores are directly correlated with the relevance of each component to the downstream task and the overall performance of the model.

Finally, the component with the highest self-attention score is selected as the scene hint $\mathbf{C}^*$. The hint encapsulates the ego agent's requirements and is then broadcast to its helper agents as a query for further processing.

3) Feature Aggregation and Prediction: The ego agent then transmits its selected key component, $\mathbf{C}^*$, to the helper agents. Based on the query communicated by the ego agent, each helper agent responds by transmitting its relevant components back. When the ego agent receives the components (comprising $\mathbf{U}$, $\mathbf{\Sigma}$, and $\mathbf{V}$) from the i-th helper agent, it first reconstructs the component as $\mathbf{F}_i = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}$. Next, $\mathbf{F}_i$ is reshaped into a tensor of dimensions $C \times H \times W$, which represents the feature map transmitted by the i-th helper agent.

After reconstructing the features from helper agents, the ego agent integrates them (i.e., $\{\mathbf{F}_i \mid i = 1, \ldots, n\}$) with its own features $\mathbf{F}$. To capture both low-level and high-level information, enhancing fusion quality and overall model performance, we adopt a multi-scale feature fusion approach, following [8], [19], [22]. By feeding $\mathbf{F}$ and $\mathbf{F}_i$ $(i = 1, \ldots, n)$ into a fusion network, the ego agent obtains a multi-scale fused feature.

The multi-scale fusion module integrates features at different scales to capture both fine-grained and high-level information. To achieve this, we adopt a series of ResNet layers [23], denoted as $\mathcal{R}_l(\cdot)$ to progressively downsample the features by a factor of 2 at each scale l, where $l \in [1, 3]$. Given the input features $\mathbf{F}^{(0)}$ and $\mathbf{F}_i^{(0)}$ from the ego agent and i-th helper agent, respectively, the encoded features at scale l are obtained through $\mathbf{F}^{(l)} = \mathcal{R}_l(\mathbf{F}^{(l-1)})$ and $\mathbf{F}_i^{(l)} = \mathcal{R}_l(\mathbf{F}_i^{(l-1)})$. At each scale l, maximum is applied to the encoded features $\mathbf{F}^{(l)}$ and $\mathbf{F}_i^{(l)}$, allowing the model to emphasize the most salient features for fusion. The fused features $\hat{\mathbf{F}}^{(l)}$ at each scale are then upsampled to the shape of $\mathbf{F}^{(0)}$. Finally, the upsampled features from all scales are concatenated along the feature dimension to produce the final fused feature vector $\hat{\mathbf{F}}$ for the decoder: $\hat{\mathbf{F}} = Cat([\mathbf{F}^{(1)}, \ldots, u_l(\mathbf{F}_i^{(l)})])$, where $u_l(\cdot)$ is the upsampling operator for the l-th scale, and $Cat(\cdot)$ is a concatenation operator along the feature dimension.

Finally, the ego agent utilizes the fused features to achieve the 3D detection results through prediction.

C. Helper Agent

As previously mentioned, the helper agent side primarily consists of two modules: FED and Components-of-Interest Selection (CIS). Since the FED module is the same as that for the ego agent, it is used to derive the component set based on the raw point cloud data observed by the helper agents. Therefore, we focus on the CIS module here.

1) Components-of-Interest Selection: The primary objective of the CIS module is to identify and select the most critical components from each helper agent for transmission to the ego agent, thereby enhancing the ego agent's 3D perception accuracy.

This involves two key aspects: first, selecting relevant components that in response to the query $\mathbf{C}^*$ from the ego agent; and second, selecting components that each helper agent itself considers significant, in order to compensate for the ego agent's observation loss due to line-of-sight obstructions. To achieve these two selection processes, the Cross-Attention Weighting (CAW) and Self-Attention Weighting (SAW) modules are proposed.

Specifically, we consider the i-th helper agent as an example. For the CAW module, first, for the helper agent's top-k components, we apply the same Global Max Pooling strategy introduced in Subsection. IV-B2 to obtain the corresponding component matrix $\mathbf{K} \in \mathbb{R}^{k \times C}$. Similarly, we perform Global Max Pooling on the hint $\mathbf{C}^*$ from ego agent, resulting in a C-dimensional vector $\hat{\mathbf{c}}$, which can be viewed as a "query".

This query vector, along with the matrix $\mathbf{K}$, is then fed into CAW. This CAW module is designed based on Multi-Head Cross-Attention (MHCA) [21] and the process of CAW is outlined in the following equations:

$$\begin{aligned} \mathrm{CAW}(\hat{\mathbf{c}}, \mathbf{K}) &= \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)\mathbf{M}^O, \\ \mathrm{head}_s &= \mathrm{Attn}(\mathbf{M}_s^Q \hat{\mathbf{c}}, \mathbf{M}_s^K \mathbf{K}) \\ &= \mathrm{softmax}\left(\frac{\mathbf{M}_s^Q \hat{\mathbf{c}} (\mathbf{M}_s^K \mathbf{K})^\intercal}{\sqrt{C}}\right), \end{aligned} \tag{6}$$

where $\mathbf{M}^O$, $\mathbf{M}_s^Q$ and $\mathbf{M}_s^K$ for $s = 1, ...h$ are trainable parameters. h is the number of heads. The CAW module computes the relevance of each component of the helper agent to the ego agent's query ($\hat{\mathbf{c}}$) and outputs a cross-attention score for each component. Using a *Relevance threshold* α, it selects the components that are most responsive to the query $\hat{\mathbf{c}}$.

The SAW module within the CIS module, it is largely consistent with the one employed on the ego agent side (Subsection. IV-B2). The main difference lies in the source of input data, which, in this case, consists of the helper agent's components.

Additionally, the output of the SAW is processed to select a subset of components deemed important by the helper agent itself, based on a given *Significance threshold* β. These selected components represent the information that the helper agent considers the most critical for transmission.

After obtaining the components through CAW and SAW, the helper agent first merges them by union operation. However, the merged components may exceed the limited real-time bandwidth constraints during transmission. In such cases, components are prioritized based on their attention scores and iteratively remove the least important ones until the limited bandwidth requirements are met.

Finally, the helper agent transmits the selected r components to the ego agent. The corresponding left singular vectors ($\mathbf{U}$), right singular vectors ($\mathbf{V}$), and singular values ($\mathbf{\Sigma}$) of the selected components are concatenated into matrices to form compact representations for efficient transmission. Specifically, the helper agent transmits the left singular vectors $\mathbf{U}$ and scaled version of the right singular vectors $\mathbf{V}'$, where the singular values $\mathbf{\Sigma}$ are pre-multiplied into $\mathbf{V}$. For a feature of size $C \times H \times W$, this approach reduces the communication volume from $O(C \cdot H \cdot W)$ to $O(r(C + H \cdot W))$. Despite the reduction in transmitted data, the ego agent can still reconstruct a high-quality approximation of the original feature, achieving strong performance with minimal communication cost.

V. Evaluation

A. Experimental Setup

1) Dataset: We employ three datasets for comprehensive evaluation as follows:

- **DAIR-V2X** [13]: DAIR-V2X is the a real-world dataset designed for vehicle-to-infrastructure (V2I) collaborative perception in autonomous driving. It captures realistic V2I perception scenarios and comprises $38,845$ frames of 3D point clouds, each involving both a vehicle and a roadside infrastructure unit. The dataset incorporates enhanced annotations from CoAlign [19], enabling 360° detection beyond the coverage of vehicle-mounted cameras.
- **OPV2V** [14]: OPV2V is a simulated dataset designed for vehicle-to-vehicle (V2V) collaborative perception, constructed via simulation using OpenCDA [17] and CARLA [18]. It contains $11,464$ frames of annotated 3D point clouds, with approximately $230,000$ 3D bounding boxes for detection tasks [14]. Each frame includes between 2 to 7 vehicles.
- **CoPeSet:** We also construct a point cloud dataset, named CoPeSet, to evaluate model performance under complex scenarios characterized by diverse environmental layouts. The dataset is generated using the CARLA [18] simulator and comprises four representative driving scenarios: *town*, *urban*, *rural*, and *highway*.
  Specifically, the town scenario contains a mixture of residential and commercial buildings; the urban scenario represents a larger metropolitan area featuring roundabouts and major intersections; the rural scenario emulates a countryside environment with narrow roads, cornfields, barns, and sparse traffic signals; and the highway scenario depicts multi-lane highways with numerous entrances and exits. For each scenario, we place a fixed number of 25 vehicles, a subset of which are equipped with virtual LiDAR sensors. The LiDAR sensor, equipped with 32 channels and scanning at 10 frames per second, generates

point clouds with an upper FOV of 2° and a lower FOV of -25°.

In total, the dataset contains $1,800$ frames of 3D point clouds, captured simultaneously from varying numbers of vehicles in each scenario, ranging from 1 to 9. Importantly, CoPeSet is used solely as a test set. No models are fine-tuned on this dataset, enabling a rigorous evaluation of the generalization capabilities of all methods.

2) Metrics: The evaluation is conducted based on the following three metrics:

- **Detection Performance**: The 3D detection performance is assessed using the Average Precision (AP) metric at specified Intersection-over-Union (IoU) thresholds, specifically AP@0.5 and AP@0.7, as delineated in [13]. Specifically, AP@0.5 refers to the AP metric evaluated at an IoU threshold of 0.5, where detection results are considered correct if the predicted bounding box overlaps with the ground truth by at least 50%.
- **Communication Cost**: The communication cost consists of two parts: the volume of queries issued by the ego agent L_q, and the volume of responses sent from each helper agent back to the ego agent L_r. The total communication cost is quantified as $(L_q + L_r) \times n$, where n represents the number of helper agents. We report the average communication cost of all transmitted features across all cases in the test dataset.
- **Response Time**: We also evaluate the response time of a candidate scheme to ensure its suitability for real-time applications. As the delay caused by computation dominates that of data communication when the cost of information to exchange is small, we measure the delay caused by data processing and model inference as the hint of the overall response time.

3) Compared Methods: We consider the following candidate schemes for comparison:

- **Top k.** Top k scheme selects the most useful SVD components based on the singular values of the corresponding components with the remainder of the scheme the same as CoPe.
- **Where2comm [10].** Where2comm adopts the strategy of reducing the spatial dimensions of the feature, i.e., the H and W of the feature, by transmitting only the part of the feature where objects are detected.
- **V2X-ViT [8].** V2X-ViT mentions reducing the number of feature channels C as a way to decrease communication cost.
- **SCOPE [12].** SCOPE leverages spatio-temporal awareness to achieve enhanced performance in 3D object detection.

4) System Implementation: We develop our method based on the OpenCOOD codebase [14]. For the LiDAR-based 3D object detection tasks, we adopt the backbone from existing work PointPillar [1] as the feature extractor. This choice is consistent with prior work in collaborative perception, including Where2comm [10] and CoAlign [19]. All experiments are conducted using the PyTorch framework. Model optimization is performed using the Adam optimizer [24]. For both the DAIR-V2X and OPV2V datasets, the training configurations include a batch size of 4 and a total of 30 training epochs. The initial learning rate is set to $1\text{e}{-3}$ and follows a step decay schedule, with the rate reduced by a factor of 0.1 every 10 epochs. This schedule is designed to facilitate stable convergence and improve generalization performance.

The *Relevance threshold* α for Cross-Attention Weighting (CAW) and the *Significance threshold* β for Self-Attention Weighting (SAW) in Components-of-Interest Selection (CIS) are both set to $\alpha = \beta = 0.1$ for the DAIR-V2X and OPV2V datasets. These threshold values are selected based on parameter configuration experiments, as detailed in *Subsection:* V-B. We set the number of selected components to $k = 8$, based on empirical studies detailed in *Subsection:* III-B, in order to filter out components associated with negligible singular values. Additionally, we configure the multi-head attention mechanism with $h = 8$ attention heads to ensure sufficient model capacity for learning diverse spatial interactions. The performance of these configurations is evaluated on the validation set of the DAIR-V2X dataset and the test sets of both the OPV2V dataset and the proposed CoPeSet.

B. Parameter Configuration

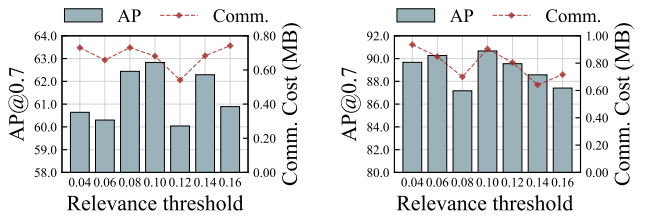
These experiments are conducted to identify the optimal threshold values for the hyper-parameters used in the Self-Attention Weighting (SAW) and Cross-Attention Weighting (CAW) within the Components-of-Interest Selection (CIS) module. The evaluation is carried out on both the DAIR-V2X and OPV2V datasets.

Relevance threshold α: We evaluate various thresholds for the CAW module during training on both the DAIR-V2X and OPV2V datasets. Different thresholds are tested during training, and the results, shown in Fig. 3(a) and Fig. 3(b), illustrate the performance variations across models on the DAIR-V2X and OPV2V datasets, respectively. The bars in the figures represent detection accuracy, while the brown dashed lines represents the corresponding communication cost. To achieve a balance between high detection accuracy and low communication cost, we determine the optimal *Relevance threshold* in CAW to be $\alpha = 0.1$ for both DAIR-V2X and OPV2V datasets.

Significance threshold β: We further evaluate the threshold settings for the SAW module. The results, as shown in Fig. 4(a) and Fig. 4(b), demonstrate the performance of various thresholds on the DAIR-V2X and OPV2V datasets, respectively. Given our criteria favoring models that couple high accuracy with low communication cost, we determine the optimal *Significance threshold* in SAW to be $\beta = 0.1$ for both DAIR-V2X and OPV2V datasets.

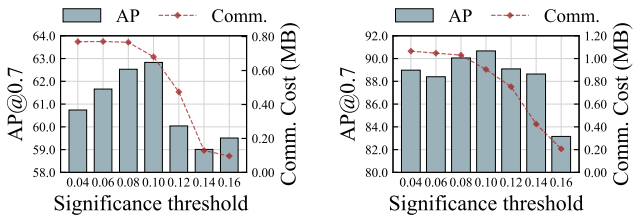
C. Ablation Study

1) Effectiveness of Core Modules: We conduct an ablation study to evaluate the effectiveness of the Multi-Scale Fusion module, the proposed Self-Attention Weighting (SAW) module



(a) DAIR-V2X (b) OPV2V

Fig. 3: Performance evaluation of the CAW module under different *Relevance threshold* values on both DAIR-V2X and OPV2V.



(a) DAIR-V2X (b) OPV2V

Fig. 4: Performance evaluation of the SAW module under different *Significance threshold* values on both DAIR-V2X and OPV2V.

TABLE I: Ablation study of the proposed SAW and CAW on DAIR-V2X and OPV2V.

Top k	Multi-Scale Fusion	SAW	CAW	DAIR-V2X				OPV2V			
				Comm(MB)	AP@0.3	AP@0.5	AP@0.7	Comm(MB)	AP@0.3	AP@0.5	AP@0.7
✓				(0.77)*n	77.40	73.37	56.79	(1.08)*n	95.63	94.78	86.01
✓	✓			(0.77)*n	77.80	73.59	59.03	(1.08)*n	95.73	95.05	87.23
✓	✓	✓		(0.68)*n	82.01	76.95	61.45	(0.85)*n	96.50	95.60	88.09
✓	✓	✓	✓	(0.69)*n	82.31	77.61	62.83	(0.90)*n	97.07	96.51	90.67

and Cross-Attention Weighting (CAW) module. Top k with single-scale fusion is adopt as the baseline. The results, summarized in Table I, demonstrate the contributions of each module both on the DAIR-V2X and OPV2V datasets.

First, it is obvious that Multi-Scale Fusion improve the detection performance on both the DAIR-V2X and OPV2V datasets. Furthermore, we can see that the inclusion of SAW improves detection performance while reducing communication cost. SAW enables the selection of the most significant components for transmission, resulting in a more efficient use of bandwidth. For DAIR-V2X, communication cost decreases from $(0.77)*n$ MB to $(0.68)*n$ MB, while AP@0.5 improves from 73.59 to 76.95, n is the number of helper agents. Similarly, for OPV2V, the communication cost decreases from $(1.08) * n$ MB to $(0.85) * n$ MB, with AP@0.5 increasing from 95.05 to 95.60. These results demonstrate SAW's ability to prioritize components that are critical for each helper agent itself and the downstream tasks, while balancing performance and bandwidth.

When CAW is added on top of SAW, detection performance improves further, although communication cost slightly increases. This trade-off is expected, as CAW selects additional components based on their relevance to the ego agent's query, as indicated by the transmitted $\mathbf{C}^*$. For DAIR-V2X, AP@0.5 improves from 76.95 to 77.61, and for OPV2V, AP@0.5 increases from 95.60 to 96.51. The slight increase in communication cost (from $(0.68) * n$ MB to $(0.69) * n$ MB for DAIR-V2X and from $(0.85) * n$ MB to $(0.90) * n$ MB for OPV2V) is justified by the corresponding performance gains.

D. Impact of Diverse Driving Scenarios

To evaluate the generalization capability of our proposed method under diverse driving environments, we conduct experiments across four synthetic yet representative scenarios in the CoPeSet dataset: Town, Urban, Rural, and Highways. These scenarios, constructed using the CARLA simulator, are detailed in *Subsection:* V-A1.

Figure 5 illustrates the comparative performance of different cooperative perception methods on the above scenarios, in terms of both detection accuracy and communication cost. The bars represent the 3D detection performance, while the dashed line indicates the communication cost transmitted by each helper agent to the ego agent. We report results under three IoU thresholds: AP@0.3, AP@0.5, and AP@0.7, from left to right in the figure. For all methods except Top-k and our proposed CoPe, a fixed communication cost of 8.59 MB of each agent is incurred. Across all four scenarios and both evaluation thresholds, CoPe consistently achieves superior detection accuracy while maintaining significantly lower communication cost. Notably, CoPe not only outperforms the Top-k baseline but also exceeds the performance of recent state-of-the-art methods such as Where2comm, V2X-ViT and Scope. The consistent performance of CoPe across diverse environments highlights its robustness and strong generalization capability.

E. Impact of Varying Communication Bandwidth

To assess the trade-off between communication efficiency and 3D object detection accuracy, we compare the performance of our proposed model, CoPe, with two representative methods: Where2comm [10] and V2X-ViT [8] under varying communication constraints on the DAIR-V2X and OPV2V datasets. The evaluation results are presented in Figs. 6(a) and 6(b), respectively.

This comparison yields two key insights. First, CoPe consistently outperforms both Where2comm and V2X-ViT in detection accuracy across all bandwidth settings, demonstrating its robustness and superior capability in modeling cooperative features under diverse and dynamic scenar-

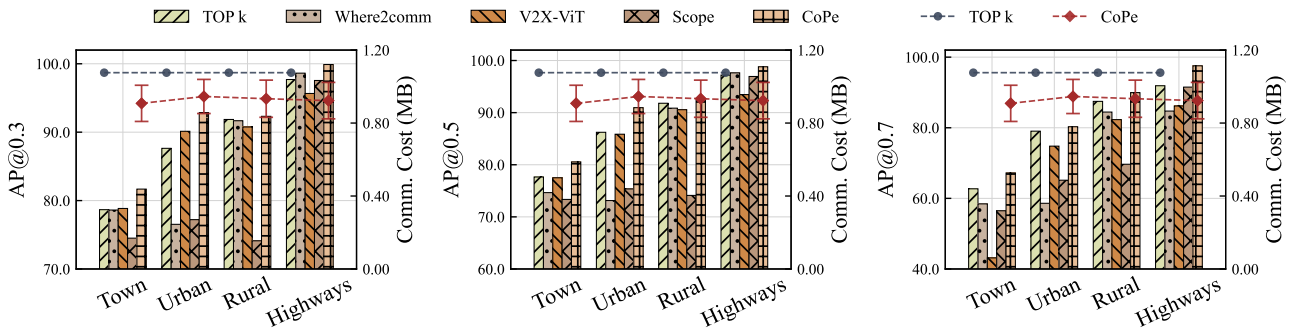
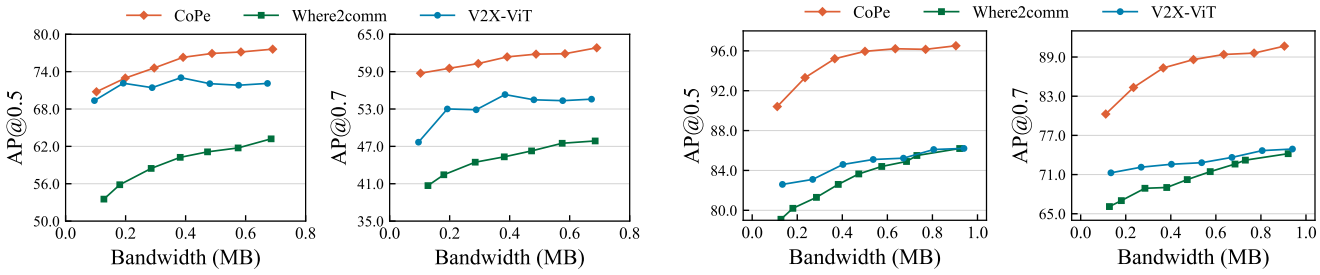


Fig. 5: 3D object detection performance and communication cost across four representative scenarios in CoPeSet: Town, Urban, Rural, and Highways. The bars represent the 3D detection performance, while the dashed line indicates the communication cost. All methods except Top-k and CoPe incur a fixed communication cost of 8.59 MB per agent.



(a) DAIR-V2X (b) OPV2V

Fig. 6: Performance comparison on DAIR-V2X and OPV2V dataset with varying communication bandwidth.

ios. Second, CoPe achieves a significantly better perception–communication trade-off. CoPe maintains high detection accuracy even under stringent communication constraints. This indicates that our model not only requires less data transmission but also does so without compromising the quality of perception, proving its efficiency and effectiveness in resource-constrained environments.

F. Response Time

The response time of CoPe and the compared models is evaluated across multiple hardware platforms, including a laptop equipped with an Nvidia RTX 2070 GPU, the Jetson TX2 NX, and the Jetson Nano. On the laptop, PyTorch is employed as the inference engine, while TensorRT [25] is utilized for runtime acceleration on the Jetson devices. All platforms operate under a Linux-based system. Table II reports the response time of all methods on the three hardware platforms. Notably, CoPe demonstrates lightweight computational characteristics, achieving a response time as low as 60ms on the Jetson TX2 NX. This result underscores the efficiency and deployability of CoPe in resource-constrained environments, making it a practical solution for real-time applications on mobile and embedded devices.

G. Overall Performance Comparison

To comprehensively evaluate the effectiveness of our proposed method, we conduct experiments on three datasets: DAIR-V2X, OPV2V, and our constructed CoPeSet. Table III presents the quantitative performance comparison across the three datasets, n is the number of helper agents.

TABLE II: Comparison of different methods in terms of response time on three mobile devices.

Method	Param.(MB)	Response Time (ms)		
		Laptop	TX2 NX	Nano
Where2comm	34.57	65	54	188
V2X-Vit	51.45	95	79	275
Scope	173.66	131	108	380
CoPe	49.41	72	60	209

As shown, CoPe significantly outperforms all compared models in both 3D detection accuracy and communication efficiency. Specifically, on DAIR-V2X, OPV2V, and CoPeSet, CoPe improves the precision of the 3D perception task at AP@0.7 by at least 12.94%, 9.93% and 13.04%, respectively, compared to the existing state-of-the-art approach [12]. In addition, it achieves a substantial reduction in communication cost by 88.78%, 89.52% and 89.28% on the corresponding datasets. These results clearly demonstrate the superiority of CoPe in balancing perception performance and communication cost.

TABLE III: Comparison of different methods in terms of Average Precision at different IOU thresholds and communication cost on DAIR-V2X, OPV2V and CoPeSet.

Method	DAIR-V2X			OPV2V			CoPeSet		
	Comm (MB)	AP@0.5	AP@0.7	Comm (MB)	AP@0.5	AP@0.7	Comm (MB)	AP@0.5	AP@0.7
Where2comm [10]	(6.15)*n	63.71	48.93	(8.59)*n	88.07	75.06	(8.59)*n	85.56	71.55
V2X-ViT [8]	(6.15)*n	72.81	54.94	(8.59)*n	86.72	74.94	(8.59)*n	88.60	65.85
SCOPE [12]	(6.15)*n	65.18	49.89	(8.59)*n	89.59	80.74	(8.59)*n	79.94	70.71
CoPe	**(0.69)***n	**77.61**	**62.83**	**(0.90)***n	**96.51**	**90.67**	**(0.92)***n	**90.68**	**83.75**

VI. Related Work

A. Collaborative Perception

Collaborative perception [13], [14], [26], [27] has been proposed as a solution to the challenges faced by single-agent perception systems, including issues such as physical occlusion [28] and limited sensor fields [29]. By enabling inter-agent communication, collaborative perception significantly enhances scene understanding and safety in multi-agent environments. Existing collaborative perception methods can be broadly categorized according to the modality of shared information into three major types: early fusion, intermediate fusion, and late fusion.

Early fusion approaches [5], [6] exchange raw sensor data, typically point clouds, between mobile agents, followed by centralized or decentralized fusion via point cloud registration (PCR) [30] or predefined localization. These methods provide rich spatial information and can achieve high 3D perception accuracy, but suffer from severe communication and computation overhead due to the high dimensionality of raw data. Late fusion methods [7] transmit only final perception results, such as 3D bounding boxes, and perform fusion through techniques such as multi-affinity graph matching. While these approaches are highly communication-efficient, they often exhibit limited improvements in perception accuracy, primarily due to the loss of semantic context and reliance on imperfect local detection outputs [8], [9]. Intermediate fusion [8], [10]–[12], [19] has recently received increasing attention as a compromise between the above two extremes. These methods share intermediate features, either directly or in compressed form, extracted from raw sensory input. By transmitting semantically rich yet compact information, intermediate fusion methods strike a more favorable balance between communication efficiency and perception accuracy.

B. Communication Efficiency

Achieving an optimal trade-off between communication efficiency and perception performance remains a fundamental challenge in intermediate fusion-based collaborative perception systems. Formally, the communication volume transmitted in the collaborative perception system can be modeled as $N \times C \times H \times W \times 4$, where N indicates the number of collaborating agents, C, H and W represent the channel, height, and width of the transmitted intermediate feature, respectively, and the constant 4 signifies that each feature value, being of type float32, occupies 4 bytes.

Several representative works have tried to reduce communication overhead. Notably, strategies such as those employed in V2VNet [31], DiscoNet [11], and V2X-ViT [8], implement 1×1 convolutional layers to reduce the channel dimension from C to C', where $C' < C$, thus lowering the communication volume. Furthermore, Where2comm [10] introduces a content-aware transmission scheme, where only spatial regions associated with detected vehicles are shared, effectively reducing the size of $H \times W$.

While the existing methods effectively reduce communication volume to an extent, they do not necessarily strike an optimal balance between performance and communication efficiency. The application of 1×1 convolutional layers, for instance, might lead to the loss of critical information. Additionally, the strategy used by Where2comm heavily depends on the detection performance of a single agent, which may not always be reliable. This approach also overlooks other semantic features in the scene that are crucial for the detection task, as it only transmits features related to vehicles.

To overcome these limitations, our work introduces a novel intermediate fusion framework based on singular value decomposition (SVD) and employ a network attention mechanism to transmit the most useful information.

VII. Conclusion

In this paper, we have proposed CoPe, a lightweight and efficient collaborative 3D perception framework for mobile agents. In CoPe, each ego agent queries the most helpful information from its neighboring agents via a lightweight network attention mechanism. By combining the ego agent's own features with the most pertinent information gathered from neighboring agents, it can achieve a comprehensive understanding of the scene, thereby enabling accurate predictions for downstream tasks. We have implemented CoPe on mobile platforms with varying computational capacities and conducted extensive experiments using both real-world and simulation datasets. The results demonstrate the effectiveness of CoPe.

Acknowledgments

This work was supported in part by the Natural Science Foundation of China (Grants No. 62432008, 62472083), Natural Science Foundation of Shanghai (Grant No. 22ZR1400200).

REFERENCES

[1] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 12 697–12 705.

[2] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4490–4499.

[3] Y. Zhou, Q. Liu, H. Zhu, Y. Li, S. Chang, and M. Guo, "Mogde: Boosting mobile monocular 3d object detection with ground depth estimation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 2033–2045, 2022.

[4] C. T. Lawyers, "How blind spots cause auto accidents," https://www.cbtrial.com/how-blind-spots-cause-auto-accidents/, n.d., accessed: 2024-12-14.

[5] 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.

[6] 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.

[7] S. Shi, J. Cui, Z. Jiang, Z. Yan, G. Xing, J. Niu, and Z. Ouyang, "Vips: Real-time perception fusion for infrastructure-assisted autonomous driving," in *Proceedings of the 28th annual international conference on mobile computing and networking*, 2022, pp. 133–146.

[8] 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.

[9] Y. Li, H. Zhu, Z. Deng, Y. Cheng, L. Zhang, S. Chang, and M. Guo, "Anole: Adapting diverse compressed models for cross-scene prediction on mobile devices," in *2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2024, pp. 613–623.

[10] Y. Hu, S. Fang, Z. Lei, Y. Zhong, and S. Chen, "Where2comm: Communication-efficient collaborative perception via spatial confidence maps," *Advances in neural information processing systems*, vol. 35, pp. 4874–4886, 2022.

[11] Y. Li, S. Ren, P. Wu, S. Chen, C. Feng, and W. Zhang, "Learning distilled collaboration graph for multi-agent perception," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29 541–29 552, 2021.

[12] K. Yang, D. Yang, J. Zhang, M. Li, Y. Liu, J. Liu, H. Wang, P. Sun, and L. Song, "Spatio-temporal domain awareness for multi-agent collaborative perception," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 23 383–23 392.

[13] 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.

[14] R. Xu, H. Xiang, X. Xia, X. Han, J. Li, and J. Ma, "Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 2583–2589.

[15] L. Miao, J. J. Virtusio, and K.-L. Hua, "Pc5-based cellular-v2x evolution and deployment," *Sensors*, vol. 21, no. 3, p. 843, 2021.

[16] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[17] R. Xu, Y. Guo, X. Han, X. Xia, H. Xiang, and J. Ma, "Opencda: An open cooperative driving automation framework integrated with co-simulation," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 1155–1162.

[18] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[19] Y. Lu, Q. Li, B. Liu, M. Dianati, C. Feng, S. Chen, and Y. Wang, "Robust collaborative 3d object detection in presence of pose errors," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 4812–4818.

[20] V. Christlein, L. Spranger, M. Seuret, A. Nicolaou, P. Král, and A. Maier, "Deep generalized max pooling," in *2019 International conference on document analysis and recognition (ICDAR)*. IEEE, 2019, pp. 1090–1096.

[21] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[22] Y. Lu, Y. Hu, Y. Zhong, D. Wang, S. Chen, and Y. Wang, "An extensible framework for open heterogeneous collaborative perception," *arXiv preprint arXiv:2401.13964*, 2024.

[23] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1492–1500.

[24] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[25] Y. Zhou and K. Yang, "Exploring tensorrt to improve real-time inference for deep learning," in *2022 IEEE 24th Int Conf on High Performance Computing & Communications; 8th Int Conf on Data Science & Systems; 20th Int Conf on Smart City; 8th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*. IEEE, 2022, pp. 2011–2018.

[26] Y. Li, D. Ma, Z. An, Z. Wang, Y. Zhong, S. Chen, and C. Feng, "V2x-sim: Multi-agent collaborative perception dataset and benchmark for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 914–10 921, 2022.

[27] R. Xu, X. Xia, J. Li, H. Li, S. Zhang, Z. Tu, Z. Meng, H. Xiang, X. Dong, R. Song *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*, 2023, pp. 13 712–13 722.

[28] Y. Han, H. Zhang, H. Li, Y. Jin, C. Lang, and Y. Li, "Collaborative perception in autonomous driving: Methods, datasets, and challenges," *IEEE Intelligent Transportation Systems Magazine*, 2023.

[29] T. Huang, J. Liu, X. Zhou, D. C. Nguyen, M. R. Azghadi, Y. Xia, Q.-L. Han, and S. Sun, "V2x cooperative perception for autonomous driving: Recent advances and challenges," *arXiv preprint arXiv:2310.03525*, 2023.

[30] Q. Liu, H. Zhu, Z. Wang, Y. Zhou, S. Chang, and M. Guo, "Extend your own correspondences: Unsupervised distant point cloud registration by progressive distance extension," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 20 816–20 826.

[31] T.-H. Wang, S. Manivasagam, M. Liang, B. Yang, W. Zeng, and R. Urtasun, "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*. Springer, 2020, pp. 605–621.