# VEHICLE-INFRASTRUCTURE COOPERATIVE 3D DETECTION VIA FEATURE FLOW PREDICTION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

In autonomous driving, 3D object detection is an important task to localize and recognize objects surrounding the ego vehicle. Cooperatively utilizing the sensor data from the ego vehicle and infrastructure can significantly expand the perception range and improve the detection ability to enhance autonomous driving safety. However, significant temporal asynchrony exists between data from ego vehicle and infrastructure in practical applications. To the best of our knowledge, no existing work in the literature could effectively solve the asynchrony problem with limited communication bandwidth and computational resources on vehicle-infrastructure devices. This work proposes a feature prediction-based framework for the vehicle-infrastructure cooperative 3D (VIC3D) object detection named Feature Flow Net (FFNet). The proposed method aims to transmit lightweight intermediate data with valuable information, with the nature of prediction, to address the problem of temporal asynchrony. We propose a self-supervised method to extract the prediction information from the video frames and train the feature flow generation model. Extensive experiments on the DAIR-V2X dataset (a large-scale real-world V2X dataset) show that FFNet establishes a new state of the art, surpassing SOTA methods by up to 5% mAP under low transmission cost. In particular, FFNet is robust to latency and can make up for almost all the performance drops caused by the temporal asynchrony within the $200ms$ delay. Code is available at anonymous-code-link.

## 1 INTRODUCTION

Although autonomous driving has achieved significant progress in recent years, it still faces vast safety challenges because of its limitation on global sensing. Utilizing information from both the ego-vehicle and the road environment via V2X communication Hobert et al. (2015); Noor-A-Rahim et al. (2022) has shown great potential to improve the autonomous driving perception ability, such as information from other autonomous driving vehicles Wang et al. (2020); Li et al. (2021); Cui et al. (2022) and infrastructure sensors Valiente et al. (2019); Liu et al. (2021); Chen et al. (2022). Among them, the vehicle-infrastructure cooperative perception is the promising direction since the infrastructure sensors are commonly installed much higher than the ego vehicles, thus having a broader field of view and alleviating the occlusion problem. 3D object detection from point clouds is one of the essential tasks in autonomous driving. This paper focuses on solving the vehicle-infrastructure cooperative 3D object detection (VIC3D) problem with point clouds as inputs.

Compared with vanilla 3D object detection Geiger et al. (2012); Caesar et al. (2020), VIC3D object detection faces more challenges. One challenge is the limited communication bandwidth between the ego-vehicle and the infrastructure devices. To meet the bandwidth requirement in communication, we need to reduce the transmission cost from infrastructure to ego vehicle while this infrastructure data is valuable for detection. Another challenge is the asynchronous timestamps between the data captured from ego-vehicle sensors and those received from the infrastructure sensors. The different sensor initialization time points and the communication delay from the infrastructure to the ego vehicle cause temporal asynchrony. Empirically, the delay commonly ranges from $100ms$ to $500ms$. The temporal asynchrony could cause serious fusion errors due to scene changing or objects moving, such as the red vehicle moving in Fig. 1. So the time compensation to align the data needs to be considered to remove the fusion error.

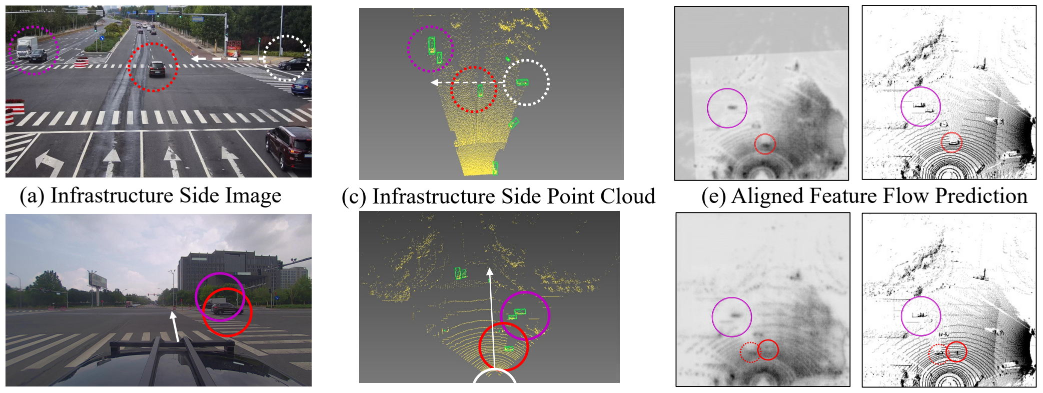

(a) Infrastructure Side Image
(c) Infrastructure Side Point Cloud
(e) Aligned Feature Flow Prediction
(b) Ego-Vehicle Side Image
(d) Ego-Vehicle Side Point Cloud
(f) Non-Aligned Feature Fusion

Figure 1: **Feature Fusion with and without Feature Prediction.** (a)-(d) Images and Point clouds Captured from the Infrastructure and Ego-Vehicle Sensors at Different Timestamps. Different timestamps are used to simulate temporal asynchrony. The vehicles in the same color circles denote the same objects. The vehicle in the white circle indicates the ego vehicle. Due to the time change, the red vehicle on the ego-vehicle side has moved some distance compared to the exact red vehicle on the infrastructure side. (e) Left: Fused Feature with Predicted Infrastructure and Ego-Vehicle Features. These two features of the red vehicle share the exact location, and there is no fusion error. Right: Point Cloud Fusion with Future Infrastructure Point Cloud and Ego-Vehicle Point Cloud. (f) Left: Fused Features with the Current Infrastructure and Ego-Vehicle Features. Two-side parts of the vehicle share different locations, resulting in a fusion error. Right: Point Cloud Fusion with the Current Infrastructure Point Cloud.

In this paper, we propose a novel feature prediction framework, Feature Flow Network (FFNet), to solve the above challenges in VIC3D object detection. We present the pipeline in Fig. 2. FFNet transmits the compressed feature flow, which integrates the time-dimension information and has the nature of prediction. By receiving the feature flow, the ego vehicle can directly predict the future feature at the timestamp of ego-vehicle data. This prediction can ensure the infrastructure and ego-vehicle data depict the scene simultaneously so that FFNet can effectively eliminate the spatial fusion error caused by temporal asynchrony. The effect of the feature prediction can be seen in Fig. 1. Moreover, the compressed feature flow maintains valuable detection information while requiring less transmission cost. It can help achieve a better performance-transmission balance than directly transmitting the detection results or the raw data. As a result, the proposed framework can effectively solve the asynchrony problem with limited transmission cost.

Since 3D annotation is very expensive, we propose a self-supervised method for FFNet training. We first train a base model without considering latency. Then we use self-supervised learning to extract temporal information from numerously infrastructure video frames and train the feature flow generator. We randomly choose a future infrastructure frame and use the feature flow to predict the feature at the timestamp of this future frame. We use the infrastructure extractor in the base model to extract the feature from this randomly assigned frame as the ground truth of the predicted feature. The training of the feature flow generator does not rely on any labeled data, so it is possible to exploit further the massive unlabelled infrastructure-side sequential data in the future.

To demonstrate the effectiveness of FFNet, we re-implement several cooperative perception methods such as V2VNet Wang et al. (2020) and DiscoNet Li et al. (2021) on the DAIR-V2X dataset Yu et al. (2022) and conduct extensive experiments on the proposed FFNet. FFNet establishes a new state of the art, surpassing SOTA methods by up to 5% mAP while with comparable transmission cost. In particular, FFNet can adapt well to different levels of communication delays on DAIR-V2X. It can make up for almost all the performance drops caused by the asynchronous data within a 200$ms$ delay.

The main contributions of our paper can be summarized as follows:

- We present a novel feature flow prediction network (FFNet) to solve the temporal asynchrony problem in VIC3D Object Detection with limited transmission cost.

- We design a self-supervised pipeline for feature flow generator training, which utilizes the history point cloud frames to predict the future feature on-the-fly.
- FFNet achieves new state-of-the-art results on the DAIR-V2X dataset, surpassing all other cooperative methods with 5% mAP improvement. Especially, FFNet behaves robustly in different latency from $100ms$ to $500ms$.

## 2 Related Work

**Ego-Vehicle-Centric 3D Detection.** Perceiving the objects, especially the 3D obstacles from the road environment, is one of the fundamental tasks in autonomous driving. The ego-vehicle-centric 3D perception can be roughly classified into three categories. a) Camera-based 3D perception. FCOS3D Wang et al. (2021) follows one-stage detector FCOS Tian et al. (2019) to directly detect the 3d bounding boxes from a single image. Li et al. (2022b); Xie et al. (2022) project the 2D image to BEV (bird-eye's view) to conduct the multi-camera joint 3D detection. b) LiDAR-based 3D perception. Zhou & Tuzel (2018); Yang et al. (2019); Lang et al. (2019) detect the objects by dividing the point cloud captured from LiDAR into voxels or pillars and extracting the feature from them. c) Multi-sensor Fusion-based 3D perception utilizes images and point clouds captured from Camera and LiDAR likeVora et al. (2020); Liu et al. (2022). Although ego-vehicle-centric 3D perception has achieved significant progress recently, it still faces enormous challenges, such as unstable detection in long-range and blind detection due to its limitation of the perception field.

**Vehicle-Infrastructure Cooperative Autonomous Driving.** With the development of V2X communication Hobert et al. (2015); Noor-A-Rahim et al. (2022), utilizing the information from the road environment has attracted much attention. Some works use information from other vehicles to broaden the perception field. V2VNet Wang et al. (2020) is a pioneering work in multi-vehicle 3D perception and provides a feature fusion framework to achieve performance-transmission trade-off. DiscoNet Li et al. (2021) applies the distillation in the feature fusion training. V2X-ViT Xu et al. (2022a) introduces the vision transformer to fuse information across on-road agents. Lei et al. (2022) propose a time compensation module for the latency. V2X-Sim Li et al. (2022a) and OPV2V Xu et al. (2022b) are two simulated datasets for multi-vehicle cooperative perception research. Some works Valiente et al. (2019); Cui et al. (2022) integrate the infrastructure data for end-to-end autonomous driving. Some works use the infrastructure to enhance the ego-vehicle-centric 3D detection ability. DAIR-V2X Yu et al. (2022) is a pioneering work in Vehicle-Infrastructure Cooperative 3D detection. It releases a large-scale real-world V2X dataset and introduces the VIC3D detection task with the early and late fusion baseline. Hu et al. (2022); Arnold et al. (2020) propose to transmit efficient data for cooperative detection. However, these works need to consider more latency. Some works like Rope3D Ye et al. (2022) and WIBAM Howe et al. (2021) focus on infrastructure-only 3D detection. This paper focuses on utilizing the infrastructure data to enhance the ego-vehicle-centric 3D detection ability with the limited transmission cost and severe latency.

**Feature Flow.** Flow is a concept originating from mathematics, which formalizes the idea of the motion of points over time Deville & Gatski (2012). It has been successfully applied to many computer vision tasks, such as optical flow Beauchemin & Barron (1995), scene flow Menze & Geiger (2015), and video recognition Zhu et al. (2017b). FlowNet Dosovitskiy et al. (2015) applies the CNN to generate the optical flow with end-to-end form. Based on FlowNet, Liu et al. (2019) generate the scene flow from the 3D point clouds. More related works about scene flow can be found in Baur et al. (2021); Behl et al. (2019). As a concept extended from optical flow Horn & Schunck (1981), feature flow describes the changing of feature maps over time, and it has been widely used in various video understanding tasks. Zhu et al. (2017b) uses both the feature and feature flow of the keyframes to predict other frames' features to speed up the inference. They use the synthetic Flying Chairs dataset Aubry et al. (2014) to train the FlowNet Dosovitskiy et al. (2015) to get a pre-trained feature flow network. Zhu et al. (2017a) proposes a flow-guided feature aggregation to improve video detection accuracy. This paper introduces the feature flow in vehicle-infrastructure cooperative perception to solve the temporal asynchrony problem.

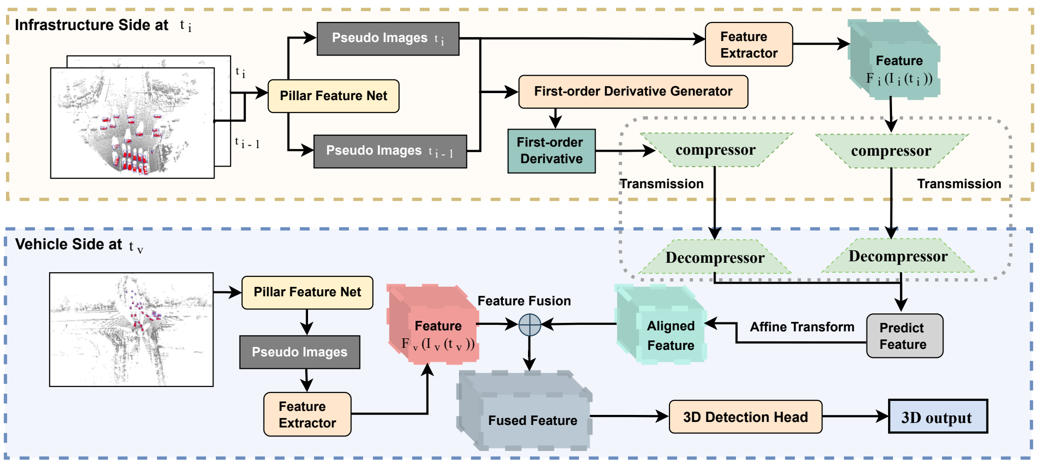


Figure 2: FFNet Overview. In infrastructure system, the feature extractor and the first-order derivative generator make up the feature flow generator. The feature flow is linearly represented with a feature and a first-order derivative as Eq. 2, and can be used to predict the future feature. The compressors focus on reducing the transmission cost. We provide the detailed configurations for FFNet in Sec. A.2 in Appendix.

## 3 METHOD

The VIC3D object detection is that the ego vehicle receives and integrates the information from infrastructure to localize and recognize the surrounding objects. The cooperative detection can significantly improve the 3D detection upper bound of the ego vehicle, such as supplementing blind spots and raising the perception horizon for the ego-vehicle. This section describes how to implement and train the proposed FFNet to solve the VIC3D object detection task.

### 3.1 VIC3D OBJECT DETECTION TASK

The VIC3D object detection can be formulated as optimizing the 3D detection performance by utilizing the infrastructure data with the limited communication bandwidth. This paper focuses on point clouds captured from LiDAR as inputs. The input of VIC3D consists of two parts:

- Ego-vehicle point cloud frame $I_v(\hat{t_v})$ captured at or before time $t_v$ (i.e. $\hat{t_v} \leq t_v$) as well as its relative pose $M_v(\hat{t_v})$, where $I_v(\cdot)$ denotes the capturing function of ego-vehicle LiDAR.
- Infrastructure point cloud frame $I_i(\hat{t_i})$ captured at or before time $t_i$ (i.e. $\hat{t_v} \leq t_i$) as well as its relative pose $M_i(\hat{t_i})$, where $I_i(\cdot)$ denotes the capturing function of infrastructure LiDAR.

VIC3D object detection has two primary goals: better detection performance and less transmission cost. Average Precision (AP) is used to measure the detection performance, and Average Byte (AB) is used to measure the transmission cost. Compared with traditional 3D object detection in autonomous driving, VIC3D object detection faces two more challenges:

- 1) There is limited communication bandwidth. For this challenge, we need to transmit as fewer data as possible while retaining the most valuable information for cooperative detection.
- 2) The communication delay and different sensor initialization cause temporal asynchrony between data from the ego vehicle and infrastructure. Directly fusing the two frames leads to the spatial fusion error. The asynchronous fusion error is illustrated in Fig. 1.

### 3.2 FEATURE FLOW NET

The implementation of FFNet is composed of the following three steps: 1) Generating the feature flow. 2) Compressing, transmitting, and decompressing the feature flow. 3) Fusing the feature flow with ego-vehicle feature to predict the detection results. The whole process is also illustrated in Fig. 2. We will introduce the FFNet training in Sec. 3.3.

**Feature Flow.** To solve the VIC3D detection task and address the above transmission cost and temporal asynchrony challenges, we should determine the proper data to transmit. There are three possible data forms for transmission: raw data like raw point clouds for early fusion, intermediate data like features for middle fusion, and object-level data for late fusion. Raw data reserves all the valuable information for detection. However, it also contains much redundant information and requires much transmission cost. Object-level data requires little transmission cost. However, it also loses much valuable information for detection. Among the three data forms, intermediate data for middle fusion is the best way to transmit valuable information while requiring affordable transmission costs. There have been some existing works Hu et al. (2022); Li et al. (2021); Lei et al. (2022) to apply middle fusion for cooperative perception. However, these intermediate data are mainly static features, which are unsuitable for predicting the future feature to solve the temporal asynchrony problem. We will further discuss this issue in Sec. 3.4 and study these issues in Sec. 4.1.

We notice a proper concept of flow originating from mathematics, which formalizes the idea of the motion of points over time Deville & Gatski (2012). It will be helpful for us to solve the temporal asynchrony. This work will apply the flow into the intermediate data as a feature flow for cooperative perception. Given the current point cloud frame $I_i(t_i)$ and infrastructure feature extractor $F_i(\cdot)$, the feature flow over the future time after $t_i$ is defined as:

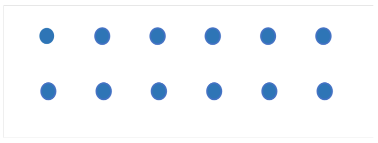
$$\widetilde{F}_i(t) = F_i(I_i(t)), t \geq t_i \tag{1}$$

If the ego-vehicle receives the feature flow, we can directly use it to predict the future infrastructure feature $\widetilde{F}_i(t_v)$ to align with the ego-vehicle timestamp $t_v$. We fuse the future infrastructure feature $\widetilde{F}_i(t_v)$ with ego-vehicle feature $F_v(I_v(t_v))$ so that we can compensate for the asynchronous time and eliminate the fusion error caused by the temporal asynchrony. Here $F_v(\cdot)$ denotes the ego-vehicle feature extractor.

**Feature Flow Generation.** Two issues need to be solved when we apply the flow into the feature flow for cooperative fusion. One is expressing and transmitting the continuous feature flow changing over time. The other is obtaining the future point cloud frames' future features after transmitting. Considering that the $t_v \rightarrow t_i$ is generally short, we use the simplest first-order approximation to estimate the feature flow with the following form to address the above two issues.

$$\widetilde{F}_i(t_i + \Delta t) \approx F_i(I_i(t_i)) + \Delta t * \widetilde{F}_i^{'}(t_i), \tag{2}$$

where the $\widetilde{F}_i^{'}(t_i)$ denotes the first-order derivative of the feature flow, and $\Delta t$ denotes a short period time in the future. That means we only need to obtain the feature $F_i(I_i(t_i))$ and the first-order derivative of the feature flow $\widetilde{F}_i^{'}(t_i)$, then we can represent the feature flow. When an ego vehicle receives the feature $F_i(I_i(t_i))$ and the first-order derivative $\widetilde{F}_i^{'}(t_i)$, we can generate the feature in arbitrary future time only with linear calculation.


(a) Static Feature

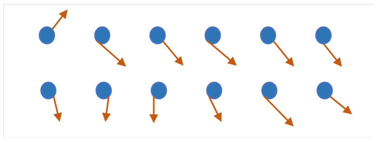

(b) Feature Flow

Figure 3: Comparison between Static Feature and Feature Flow. Feature flow involves more prediction information.

Now we explain how to generate the first-order derivative of the feature flow $\widetilde{F}_i^{'}(t_i)$. We estimate and predict the $\widetilde{F}_i^{'}(t_i)$ from the historical infrastructure frames $\{I_i(t_i - N + 1), \cdots, I_i(t_i - 1), I_i(t_i)\}$. Here we take $N$ as 2, which means we use two consecutive infrastructure frames $I_i(t_i - 1)$ and $I_i(t_i)$ to generate the first-order derivative of the feature flow. Naturally, we can also use more frames to generate more accurate estimations. We first use the Pillar Feature Net Lang et al. (2019) to convert the two point clouds to two pseudo-images of the same size. Then we concatenate the two pseudo-images and use another modified feature extractor to generate the estimated first-order derivative $\widetilde{F}_i^{'}(t_i)$. Finally, with the static feature $F_i(I_i(t_i))$ and the estimated first-order derivative $\widetilde{F}_i^{'}(t_i)$, we can represent and obtain the feature flow. Here the size of the static feature $F_i(I_i(t_i))$ and the first-order derivative $\widetilde{F}_i^{'}(t_i)$ are both $[384, 288, 288]$.

**Compression, Transmission and Decompression.** According to the above analysis, the transmission cost of feature flow is enormous, up to 384×288×288×2 floating point numbers. To further reduce the transmission and remove the redundant information, we respectively apply two compressors to compress the static $F_i(I_i(t_i))$ and the first-order derivative $\widetilde{F}_i^{'}(t_i)$ from size $[384, 288, 288]$ to size $[384/32, 288/8, 288/8]$. Here the compressors are composed of three Conv-Bn-ReLU blocks. Then we transmit the compressed feature flow and the calibration files and decompress the feature flow with two decompressors to the original size $[384, 288, 288]$ in ego vehicle. Here the decompressors are composed of three Deconv-Bn-ReLU blocks. Note that we only transmit the compressed feature flow, that is, $1/32 \times 1/8 \times 1/8$ of the original feature flow and about $1.2 \times 10^5$ bytes.

**Vehicle-infrastructure Feature Fusion.** After the ego vehicle receives the decompressed feature flow, we use this feature flow to predict the infrastructure feature to align with the ego-vehicle data at timestamp $t_v$ as

$$\widetilde{F}_i(t_v) \approx F_i(I_i(t_i)) + (t_v - t_i) * \widetilde{F}_i^{'}(t_i). \tag{3}$$

Then we transform the predicted infrastructure feature $\widetilde{F}_i(t_v)$ into the ego-vehicle coordinate system. Next, we concatenate the transformed infrastructure feature with the ego-vehicle feature and use a Conv-Bn-Relu block to fuse the concatenated feature. Ultimately, we input the fused feature into a 3D detection head to generate 3D outputs. Experimental results show that our feature flow prediction module can effectively compensate for the performance drop by the temporal asynchrony and is robust to different communication delays.

## 3.3 Feature Flow Generator Training

Manual 3D annotations, incredibly cooperative 3D annotations, are extremely expensive. By contrast, one can easily obtain unlabeled infrastructure data continuously. This part explains how we use self-supervised learning to train the feature flow generator with the infrastructure video frames.

The feature flow generator training consists of (1) the infrastructure feature extractor training and (2) the first-order derivative generator training. We first train the FFNet on the cooperative data with the cooperative annotations. In this training, we ignore the latency to get a feature fusion base model. We can get the trained infrastructure feature extractor from this feature fusion base model. Then we use the infrastructure video frames and the trained infrastructure feature extractor to teach the first-order derivative generator. These video frames may not require additional annotations. With these infrastructure video frames, we construct the training frame pairs $\mathcal{D} = \{d_{t_i,k} = (I_i(t_i - 1), I_i(t_i), I_i(t_i + k))\}$, where $I_i(t_i - 1)$ and $I_i(t_i)$ are two consecutive infrastructure point cloud frames, $I_i(t_i + k)$ is the next $k$th point cloud frame of the $I_i(t_i)$. Here the two consecutive frames are used to generate the feature flow with the start time $t_i$, $I_i(t_i + k)$ is used to create the ground truth feature for the generated feature flow. For each pair $d_{t_i,k}$ in $\mathcal{D}$:

- We input the $I_i(t_i - 1)$ and $I_i(t_i)$ into the feature flow generator to generate the feature flow, which is composed of $F_i(I_i(t_i))$ and the estimated first-order derivative $\widetilde{F}_i^{'}(t_i)$ as Eq. 2.
- We use the feature flow to predict the feature at $t_i + k$ as
$$\widetilde{F}_i(t_i + k) \approx F_i(I_i(t_i)) + |(t_i + k) - t_i| * \widetilde{F}_i^{'}(t_i). \tag{4}$$
- We use $F_i(\cdot)$ to extract the feature $F_i(I_i(t_i + k))$ from $I_i(t_i + k)$ as the ground truth feature.

Now, we use the ground truth feature $F_i(I_i(t_i + k))$ to supervise the feature flow generator training. The objective of the feature flow generator is to generate the feature flow and use it to predict the $\widetilde{F}_i(t_i + k)$ as close as possible to the $F_i(I_i(t_i + k))$. Cosine similarity is widely used to calculate the similarity of two vectors. Here we use the cosine similarity to measure the similarity between $\widetilde{F}_i(I_i(t_i + k))$ and $F_i(I_i(t_i + k))$ as

$$similarity = \frac{\widetilde{F}_i(t_i + k) \odot F_i(I_i(t_i + k))}{||\widetilde{F}_i(t_i + k)|| * ||F_i(I_i(t_i + k))||}, \tag{5}$$

where $\odot$ denotes the inner product. To train the feature flow generator, we minimize the following similarity loss

$$\mathcal{L}(\mathcal{D}, \theta) = \sum_{d_{t_i,k} \in \mathcal{D}} (1 - \frac{\widetilde{F}_i(t_i + k) \odot F_i(I_i(t_i + k))}{||\widetilde{F}_i(t_i + k)|| * ||F_i(I_i(t_i + k))||}), \tag{6}$$

where $\theta$ is the parameter of the feature flow generator, and we only update the parameters in first-order derivative generator $\widetilde{F}_i^{'}(\cdot)$.

**Remark.** We noticed that the cosine similarity is independent of the magnitudes of the two input tensors. For example, $[1, 2, 3]$ and $[2, 4, 6]$ achieve the maximal cosine similarity value $1$, but the values of the two tensors are different. Hence, we need a scale transformation to adjust the magnitudes of $\widetilde{F}_i(t_i + k)$. We take $||F_i(I_i(t_i))||_1/||\widetilde{F}_i(t_i + k)||_1$ [1] as the scale transformation.

**Algorithm 1** The training process of the first-order derivative generator in feature flow generator.

```
Input: training dataset $\mathcal{D}$, trained infrastructure feature extractor $F_i(\cdot)$
Output: Trained first-order derivative generator
1: Initialize the parameter of $\widetilde{F}_i^{'}(\cdot)$ in Eq. 2
2: for $Iteration$ from 1 to $Iteration_{max}$ do
3:     Sample the frame pair $d_{t_i,k}$ from $\mathcal{D}$
4:     Generate $F_i(I_i(t_i))$ and $\widetilde{F}_i^{'}(t_i)$ to represent the feature flow as Eq. 2
5:     Predict the feature $\widetilde{F}_i(t_i + k)$ with feature flow as Eq. 4
6:     Generate the ground truth feature $F_i(I_i(t_i + k))$
7:     Update the parameter $\theta$ with $\mathcal{L}$ in Eq. 6
8: end for
9: return The parameters of the first-order derivative generator.
```

### 3.4 Relationship to Other Existing Possible Solutions

Compared with the possible existing solutions, FFNet provides a more applicable paradigm to implement the vehicle-infrastructure cooperative perception with the following advantages:

- FFNet aims to reach the communication bandwidth requirement and provide valuable and complementary information for on-vehicle perception. Compared with early fusion, FFNet transmits the compressed intermediate data and does not require a vast transmission cost. FFNet has much valuable information for the on-vehicle perception compared with late fusion.
- FFNet can address the temporal asynchrony problem between the data captured from ego-vehicle sensors and received from the infrastructure sensors. Compared with those feature fusion methods like V2VNet Wang et al. (2020) and DiscoNet Li et al. (2021), which do not consider temporal asynchrony problem, FFNet transmits the feature flow with the nature of future prediction. The feature flow can be used to generate the future feature aligned with the ego-vehicle feature, to alleviate the fusion error caused by the temporal asynchrony. Notice that the first-order derivative in feature flow is an independent module from the static feature. We can also add this module to other feature fusion methods to achieve lower transmission costs.
- FFNet is computing-friendly and storage-friendly for vehicles with limited computing and storage resources. Because our feature flow is wholly generated on the infrastructure side, it can be directly used to predict future features with the linear computation in ego vehicles. In addition, it can be released without storage required when used up. The dropped frames will not affect our feature flow prediction because our prediction is not dependent on historical receiving. We notice that there is another possible solution to solve the temporal asynchrony problem Lei et al. (2022), which generates the feature predictor with the received historical features in the ego vehicle. But extracting the temporal information from the compressed feature could be difficult. Moreover, this solution needs memory to store the historical features and many computing resources to process these frames. More experiment comparisons are illustrated in Sec. 4.2.
- FFNet training requires much less labeled data and annotation costs. With the frame pairs sampled from infrastructure video frames, we use self-supervised learning to learn the temporal information and train the feature flow generator. The training does not rely on the labeled data, so it is possible to exploit further the massive unlabelled infrastructure-side sequential data in the future.

## 4 Experiments

In this section, we first implement FFNet and different fusion methods to solve VIC3D detection on DAIR-V2X dataset Yu et al. (2022) and compare their experiment results. Secondly, we study

[1]$|| \cdot ||_1$ denotes the L1 norm.

Table 1: **VIC3D Detection Results with FFNet and Different Fusion Methods.** "AB" denotes the average byte used to measure the transmission cost. "/" denotes that there is no latency for non-fusion methods. "-" denotes that no information is provided. FFNet outperforms non-fusion method up to 10.90% mAP@BEV (IoU=0.5) and 10.96% mAP@BEV (IoU=0.5) in $100ms$ and $200ms$ respectively. FFNet also outperforms all other fusion methods when the delay reaches $200ms$. Significantly, FFNet surpasses the early fusion more than 2% mAP@BEV (IoU=0.5) in $200ms$ latency while requiring only no more than $1/10$ transmission cost.

| Model | FusionType | Latency (ms) | mAP@3D ↑ | | mAP@BEV ↑ | | AB (Byte) ↓ |
|---|---|---|---|---|---|---|---|
| | | | IoU=0.5 | IoU=0.7 | IoU=0.5 | IoU=0.7 | |
| PointPillars Lang et al. (2019) | non-fusion | / | 48.06 | - | 52.24 | - | 0 |
| Early Fusion | early | 100 | 57.35 | - | 64.06 | - | $1.4\times10^6$ |
| TCLF Yu et al. (2022) | late | 100 | 40.79 | - | 46.80 | - | $5.4\times10^2$ |
| DiscoNet Li et al. (2021) | middle | 100 | 52.83 | 29.19 | 61.25 | 50.11 | $1.2\times10^5$ |
| V2VNet Wang et al. (2020) | middle | 100 | 52.02 | 28.54 | 60.78 | 50.02 | $1.2\times10^5$ |
| **FFNet (Ours)** | middle | 100 | 55.48 | 31.5 | **63.14**(+10.90) | 54.28 | $1.2\times10^5$ |
| Early Fusion | early | 200 | 54.63 | - | 61.08 | - | $1.4\times10^6$ |
| TCLF Yu et al. (2022) | late | 200 | 36.72 | - | 41.67 | - | $5.1\times10^2$ |
| DiscoNet Li et al. (2021) | middle | 200 | 50.76 | 28.57 | 58.20 | 48.90 | $1.2\times10^5$ |
| V2VNet Wang et al. (2020) | middle | 200 | 49.67 | 26.96 | 56.02 | 46.32 | $1.2\times10^5$ |
| **FFNet (Ours)** | middle | 200 | 55.37 | 31.66 | **63.20** (+10.96) | 54.69 | $1.2\times10^5$ |

how the FFNet solves the temporal asynchrony problem with the feature flow. Thirdly, we evaluate the FFNet in more latency to show the robustness of the FFNet. In Appendix, we also compare extracting the feature flow on different sides (infrastructure side *vs.* ego vehicle side). We develop the models based on MMDetection3D mmd (2020).

### 4.1 Comparison with Different Fusion Methods

We compare our FFNet with four different fusion methods: non-fusion *e.g.* PointPillars Lang et al. (2019), early fusion, late fusion *e.g.* TCLF Yu et al. (2022), middle fusion *e.g.* DiscoNet Li et al. (2021) and V2VNet Li et al. (2022a). We evaluate the FFNet and those different fusion methods under 100ms and 200ms latency, respectively. All detection results are measured with the KITTI Geiger et al. (2012) evaluation detection metrics: bird's eye view (BEV) mAP and 3D mAP with 0.5 IoU and 0.7 IoU, respectively. Furthermore, only objects located at the designed rectangular area [0, -39.12, 100, 39.12] are considered in the metrics. We provide more implementation details of FFNet and those fusion methods in the Appendix. We present the evaluation results in Tab. 1.

**Result Analysis.** Firstly, compared with the non-fusion method, our FFNet surpasses PointPillars 10.90 % BEV-mAP (IoU=0.5) and 10.96% BEV-mAP (IoU=0.5) in $100ms$ and $200ms$ latency respectively. This result shows that utilizing infrastructure data can improve detection performance. Secondly, although late fusion with transmitting the detection results requires little transmission cost, BEV-mAP (IoU=0.5) of TCLF is much lower than that of FFNet up to 21.53% in $200ms$ latency. Thirdly, compared with early fusion methods, FFNet achieves similar detection performance in $100ms$ latency and outperforms more than 2% mAP in $200ms$ latency, while it only requires no more than 1/10 transmission cost. Fourthly, our FFNet achieves the best detection performance with the exact transmission cost as the middle fusion methods. For example, our FFNet surpasses DiscoNet 1.89% BEV-mAP (IoU=0.5) and 5.0% BEV-mAP (IoU=0.5) in $100ms$ and $200ms$ latency, respectively. In summary, our FFNet achieves new SOTA on DAIR-V2X in $200ms$ latency while consuming little transmission cost.

### 4.2 Ablation Study

We conduct more extensive experiments to demonstrate that the feature flow module plays a vital role in solving the temporal asynchrony problem and that FFNet is robust to different latency.

**Feature prediction can well solve temporal asynchrony.** We first evaluate the FFNet in $0ms$ latency and $200ms$ latency on DAIR-V2X. Here $0ms$ latency means no temporal asynchrony between infrastructure data and ego-vehicle data. Then we remove the prediction module from FFNet, called FFNet (without prediction). We evaluate the FFNet (without prediction) in $0ms$ latency and $200ms$ latency. Since FFNet (without prediction) does not need to transmit the first-order derivative of the feature flow, it only requires half of the transmission cost of FFNet. To make a fair comparison, we

Table 2: **Comparison between with and without Feature Prediction.** Compared with no prediction models, FFNet with feature prediction has much less performance drop when there is latency.

| Model | Latency (ms) | mAP@3D ↑ | | mAP@BEV ↑ | | AB (Byte) ↓ |
|---|---|---|---|---|---|---|
| | | IoU=0.5 | IoU=0.7 | IoU=0.5 | IoU=0.7 | |
| FFNet | 0 | 55.81 | 30.23 | 63.54 | 54.16 | $1.2\times10^5$ |
| FFNet (without prediction) | 0 | 55.81 | 30.23 | 63.54 | 54.16 | $6.2\times10^4$ |
| FFNet-V2 (without prediction) | 0 | 55.78 | 30.22 | 64.23 | 55.00 | $1.2\times10^5$ |
| FFNet | 200 | 55.37 | 31.66 | 63.20 (-0.34) | 54.69 | $1.2\times10^5$ |
| FFNet (without prediction) | 200 | 50.27 | 27.57 | 57.93 (-5.61) | 48.16 | $6.2\times10^4$ |
| FFNet-V2 (without prediction) | 200 | 49.90 | 27.33 | 58.00 (-6.23) | 48.22 | $1.2\times10^5$ |

also train another FFNet, called FFNet-V2, that compresses the feature flow from (384, 288, 288) to (384/16, 288/8, 288/8). So the FFNet-V2 (without prediction) has the exact transmission cost as FFNet. We also evaluate FFNet-V2 (without prediction) in $0ms$ to and $200ms$ latency. We present the evaluation results in Tab. 2.

As Tab. 2 shows, FFNet (without prediction) and FFNet-V2 (without prediction) both have a significant performance drop when there is $200ms$ latency. For example, FFNet (without prediction) has $5.61\%$ BEV-mAP (IoU=0.5) drop in $200ms$ latency compared to $0ms$ latency. Although FFNet-V2 (without prediction) performs slightly better than FFNet and FFNet (without prediction) in $0ms$ latency, FFNet significantly surpasses the FFNet-V2 (without prediction) in $200ms$ latency. The experiment results show that temporal asynchrony can cause a serious performance drop for middle fusion methods that only transmit the static feature. And our feature prediction module can effectively compensate for the performance drop caused by temporal asynchrony.

**FFNet is robust to different latency.** We further evaluate the FFNet, FFNet (without prediction), and FFNet-V2 (without prediction) in more latency cases, from $100ms$ to $500ms$. We present the experiment results in Fig. 4. In Fig. 4, FFNet (without prediction) and FFNet-V2 (without prediction) both have continuous performance drops as they increase from $100ms$ to $500ms$. Especially, FFNet (without prediction) and FFNet-V2 (without prediction) have $9.38\%$ BEV-mAP (IoU=0.5) drop and $9.76\%$ BEV-mAP (IoU=0.5) drop respectively in $500ms$ latency. In comparison, FFNet has little performance drop within $200ms$ latency and only has a $4.39\%$ BEV-mAP (IoU=0.5) drop. This result effectively demonstrates that our FFNet is robust to different latency.

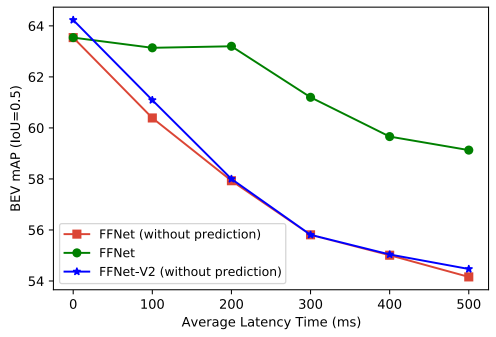


Figure 4: Average latency vs. detection performance. The green curve denotes the evaluation results of the FFNet, and it behaves robustly in different latency, compared with no prediction models.

## 5 Conclusion

This paper proposes a novel prediction framework called FFNet to solve the VIC3D detection. Compared with previous middle fusion approaches that only transmit the static feature map, the FFNet transmits the feature flow with the nature of future prediction. With the feature flow, the proposed method can predict future features to align with ego-vehicle features, effectively alleviating the spatial fusion error caused by temporal asynchrony. This paper also designs a novel self-supervised learning to exploit multiple infrastructure-side unlabelled video frames. The experimental results on DAIR-V2X show that FFNet achieves the state-of-art and outperforms other methods up to $5\%$ mAP in $200ms$ delay while consuming no more than 1/10 transmission cost compared with transmitting raw data. Primarily, FFNet performs robustly in different latency from $100ms$ to $500ms$.

## REFERENCES

MMDetection3D: OpenMMLab next-generation platform for general 3D object detection, 2020.

Eduardo Arnold, Mehrdad Dianati, Robert de Temple, and Saber Fallah. Cooperative perception for 3d object detection in driving scenarios using infrastructure sensors. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

Mathieu Aubry, Daniel Maturana, Alexei A Efros, Bryan C Russell, and Josef Sivic. Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3762–3769, 2014.

Stefan Andreas Baur, David Josef Emmerichs, Frank Moosmann, Peter Pinggera, Björn Ommer, and Andreas Geiger. Slim: Self-supervised lidar scene flow and motion segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13126–13136, 2021.

Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM computing surveys (CSUR)*, 27(3):433–466, 1995.

Aseem Behl, Despoina Paschalidou, Simon Donné, and Andreas Geiger. Pointflownet: Learning representations for rigid motion estimation from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7962–7971, 2019.

Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11621–11631, 2020.

Jing Chen, Qichao Wang, Harry H Cheng, Weiming Peng, and Wenqiang Xu. A review of vision-based traffic semantic understanding in itss. *IEEE Transactions on Intelligent Transportation Systems*, 2022.

Jiaxun Cui, Hang Qiu, Dian Chen, Peter Stone, and Yuke Zhu. Coopernaut: End-to-end driving with cooperative perception for networked vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 17252–17262, 2022.

Michel Deville and Thomas B Gatski. *Mathematical modeling for complex fluids and flows*. Springer Science & Business Media, 2012.

Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 2758–2766, 2015.

Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 3354–3361. IEEE, 2012.

Laurens Hobert, Andreas Festag, Ignacio Llatser, Luciano Altomare, Filippo Visintainer, and Andras Kovacs. Enhancements of v2x communication in support of cooperative autonomous driving. *IEEE communications magazine*, 53(12):64–70, 2015.

Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3): 185–203, 1981.

Matthew Howe, Ian Reid, and Jamie Mackenzie. Weakly supervised training of monocular 3d object detectors using wide baseline multi-view traffic camera data. In *The British Machine Vision Conference (BMVC)*, 2021.

Yue Hu, Shaoheng Fang, Zixing Lei, Yiqi Zhong, and Siheng Chen. Where2comm: Communication-efficient collaborative perception via spatial confidence maps. *arXiv preprint arXiv:2209.12836*, 2022.

Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds, 2019.

Zixing Lei, Shunli Ren, Yue Hu, Wenjun Zhang, and Siheng Chen. Latency-aware collaborative perception. In *European Conference on Computer Vision*. Springer, 2022.

Yiming Li, Shunli Ren, Pengxiang Wu, Siheng Chen, Chen Feng, and Wenjun Zhang. Learning distilled collaboration graph for multi-agent perception. *Advances in Neural Information Processing Systems*, 34, 2021.

Yiming Li, Dekun Ma, Ziyan An, Zixun Wang, Yiqi Zhong, Siheng Chen, and Chen Feng. V2x-sim: Multi-agent collaborative perception dataset and benchmark for autonomous driving. *IEEE Robotics and Automation Letters*, 7(4):10914–10921, 2022a.

Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. *arXiv preprint arXiv:2203.17270*, 2022b.

Shaoshan Liu, Bo Yu, Jie Tang, and Qi Zhu. Towards fully intelligent transportation through infrastructure-vehicle cooperative autonomous driving: Challenges and opportunities. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 1323–1326. IEEE, 2021.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pp. 21–37. Springer, 2016.

Xingyu Liu, Charles R. Qi, and Leonidas J. Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. *arXiv preprint arXiv:2205.13542*, 2022.

Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3061–3070, 2015.

Md Noor-A-Rahim, Zilong Liu, Haeyoung Lee, Mohammad Omar Khyam, Jianhua He, Dirk Pesch, Klaus Moessner, Walid Saad, and H Vincent Poor. 6g for vehicle-to-everything (v2x) communications: Enabling technologies, challenges, and opportunities. *Proceedings of the IEEE*, 2022.

Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9627–9636, 2019.

Rodolfo Valiente, Mahdi Zaman, Sedat Ozer, and Yaser P Fallah. Controlling steering angle for cooperative self-driving vehicles utilizing cnn and lstm-based deep networks. In *2019 IEEE intelligent vehicles symposium (IV)*, pp. 2423–2428. IEEE, 2019.

Sourabh Vora, Alex H Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4604–4612, 2020.

Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Fcos3d: Fully convolutional one-stage monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 913–922, 2021.

Tsun-Hsuan Wang, Sivabalan Manivasagam, Ming Liang, Bin Yang, Wenyuan Zeng, and Raquel Urtasun. V2vnet: Vehicle-to-vehicle communication for joint perception and prediction. In *European Conference on Computer Vision*, pp. 605–621. Springer, 2020.

Enze Xie, Zhiding Yu, Daquan Zhou, Jonah Philion, Anima Anandkumar, Sanja Fidler, Ping Luo, and Jose M Alvarez. Mˆ 2bev: Multi-camera joint 3d detection and segmentation with unified birds-eye view representation. *arXiv preprint arXiv:2204.05088*, 2022.

Runsheng Xu, Hao Xiang, Zhengzhong Tu, Xin Xia, Ming-Hsuan Yang, and Jiaqi Ma. V2x-vit: Vehicle-to-everything cooperative perception with vision transformer. *arXiv preprint arXiv:2203.10638*, 2022a.

Runsheng Xu, Hao Xiang, Xin Xia, Xu Han, Jinlong Liu, and Jiaqi Ma. Opv2v: An open benchmark dataset and fusion pipeline for perception with vehicle-to-vehicle communication. *ICRA*, 2022b.

Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1951–1960, 2019.

Xiaoqing Ye, Mao Shu, Hanyu Li, Yifeng Shi, Yingying Li, Guangjie Wang, Xiao Tan, and Errui Ding. Rope3d: The roadside perception dataset for autonomous driving and monocular 3d object detection task. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21341–21350, 2022.

Haibao Yu, Yizhen Luo, Mao Shu, Yiyi Huo, Zebang Yang, Yifeng Shi, Zhenglong Guo, Hanyu Li, Xing Hu, Jirui Yuan, and Zaiqing Nie. Dair-v2x: A large-scale dataset for vehicle-infrastructure cooperative 3d object detection. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2022.

Yin Zhou and Oncel 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*, pp. 4490–4499, 2018.

Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 408–417, 2017a.

Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2349–2358, 2017b.

## A Other Experiment Details

### A.1 DAIR-V2X Dataset

DAIR-V2X Yu et al. (2022) is a large-scale vehicle-infrastructure dataset, and all frames are captured from the real world and equipped with 3D annotations. There are more than 100 scenes and 18000 data pairs. The data pair are all captured from the infrastructure Camera, infrastructure LiDAR, ego-vehicle Camera, and ego-vehicle LiDAR simultaneously when the ego vehicle passes through the intersection. It also provides the cooperative 3D annotation with infrastructure view and ego-vehicle view for 9311 pairs, and each object is also labeled with the category from (*Car*, *Bus*, *Truck*, and *Van*). The dataset is split into *train/val/test* as 5:2:3. All the models are evaluated on *val* part.

### A.2 Feature Flow Network Setting

The feature flow network is mainly composed of six parts. 1) The feature flow prediction module. The infrastructure PFNet (Pillar Feature Net) shares the same architecture as PointPillars Lang et al. (2019). The x, y, and z ranges of the input point cloud are [(0, 92.16), (-46.08, 46.08), (-3, 1)] meters, respectively. The voxel size of x, y, and z are [0.16, 0.16, 4] meters, respectively. The output shape of the pseudo-images is (64, 576, 576). The infrastructure feature extractor $F_i(\cdot)$ and the estimated first-order derivative generator $\widetilde{F}_i^{'}(\cdot)$ share the same Backbone and FPN as SECOND Yan et al. (2018). The output shape of the feature and estimated first-order derivative are both [384, 288, 288]. 2) The compressor and decompressor. The compressor has four convolutional blocks with strides (2, 1, 2, 2) to compress the features from (384, 288, 288) to (384/32, 288/8, 288/8). The decompressor has three deconvolutional blocks with strides (2, 2, 2) to decompress the features to the original size. 3) Affine transform module. We implement the affine transform with the affine_grid function supported in Pytorch. We ignore the rotation around the x-y plane. 4) Feature fusion module. The fusion module is a $3 \times 3$ convolutional block with stride 1 to compress the concatenate feature from (768, 288, 288) to (384, 288, 288). 5) Ego-vehicle feature extractor. This extractor has the same configuration as the infrastructure PFNet and feature extractor. 6) 3D detection head. We use the Single Shot Detector (SSD) Liu et al. (2016) to generate the 3D outputs. The anchor has a width, length, and height of (1.6, 3.9, 1.56) meters with a z-center of -1.78 meters. Positive and negative thresholds of matching are 0.6 and 0.45, respectively.

We train the feature fusion base model on the training part of the DAIR-V2X for 40 epochs. The learning rate is 0.001, and the weight decay is 0.01. We randomly select 11037 frames from the training part of DAIR-V2X and randomly set $k$ from [1, 2] to form $\mathcal{D}$, the explanation of $k$ and $\mathcal{D}$ can be seen in Sec. 3.3. We use the trained feature fusion base model to pretrain FFNet. We train the feature flow generator on $D_u$ for 10 epochs, with a 0.001 learning rate and 0.01 weight decay. We implement all the training and evaluation with NVIDIA GeForce RTX 3090 GPU.

### A.3 DiscoNet and V2VNet Setting

DiscoNet Li et al. (2021) and V2VNet Wang et al. (2020) were initially applied to multi-vehicle cooperative perception methods. They transmit and receive the static feature for feature fusion. To apply them to VIC3D detection and compare them with our FFNet, we made three modifications: 1) Removing multi-vehicle selection and keeping only one vehicle setting. 2) Reuse the architecture of the feature fusion base model as much as possible. 3) Compressing the features from (384, 288, 288) to (384, 288/8, 288/8) to keep the comparative transmission cost with FFNet. We also train the two models on the training part of the DAIR-V2X dataset for 40 epochs. The other training configurations are the same as FFNet.

## B Ablation Study

This section discusses the effect of extracting the feature flow on different sides (infrastructure side *vs.* ego vehicle side).

**Generating feature flow on infrastructure is better than on ego vehicle.** To compare the effect of extracting feature flow between in infrastructure and ego vehicle, we train a modified FFNet called FFNet-V. The FFNet-V uses the consecutive features $F_i(I_i(t_i - 1))$ and $F_i(I_i(t_i - 1))$ received from

Table 3: **Extracting Feature Flow on Infrastructure side *vs.* on Ego-Vehicle Side.** FFNet-V denotes the model that extracts the feature flow on vehicle. FFNet-V (Same-TC) denotes the FFNet-V which has the exact transmission cost as FFNet. "AB" denotes the average byte used to measure the transmission cost. "SCC" indicates the storage cost complexity for the ego vehicle to store the past frames, "CCC" indicates the computing cost complexity for ego vehicle to extract the feature flow, "N" indicates the number of historical structures to be used. The SCC of FFNet is O(1) because it does not need extra historical frames on ego vehicle. At the same time, the SCC of extracting flow on ego vehicle is O(N) because extracting flow on ego vehicle needs past frames received from infrastructure. Moreover, FFNet achieves better detection performs, and this advantage becomes more pronounced (+3% mAP) when latency increases to 300$ms$.

| Model | Latency (ms) | mAP@3D ↑ | | mAP@BEV ↑ | | AB(Byte) ↓ | SCC ↓ | CCC ↓ |
|---|---|---|---|---|---|---|---|---|
| | | IoU=0.5 | IoU=0.7 | IoU=0.5 | IoU=0.7 | | | |
| FFNet-V | 100 | 53.21 | 28.43 | 61.50 | 50.50 | $6.2\times10^4$ | O(N) | O(N) |
| FFNet-V (Same-TC) | 100 | 53.17 | 28.45 | 62.44 | 51.68 | $1.2\times10^5$ | O(N) | O(N) |
| **FFNet (Ours)** | 100 | 55.48 | 31.50 | **63.14** (+0.7) | 54.28 | $1.2\times10^5$ | O(1) | O(1) |
| FFNet-V | 300 | 50.81 | 28.45 | 57.75 | 49.62 | $6.2\times10^4$ | O(N) | O(N) |
| FFNet-V (Same-TC) | 300 | 50.5 | 28.25 | 58.02 | 50.03 | $1.2\times10^5$ | O(N) | O(N) |
| **FFNet (Ours)** | 300 | 53.46 | 30.42 | **61.20** (+3.18) | 52.44 | $1.2\times10^5$ | O(1) | O(1) |
| FFNet-V | 500 | 49.93 | 28.63 | 56.42 | 48.87 | $6.2\times10^4$ | O(N) | O(N) |
| FFNet-V (Same-TC) | 500 | 49.98 | 27.7 | 56.99 | 49.55 | $1.2\times10^5$ | O(N) | O(N) |
| **FFNet (Ours)** | 500 | 52.08 | 30.11 | **59.13** (+2.14) | 51.70 | $1.2\times10^5$ | O(1) | O(1) |

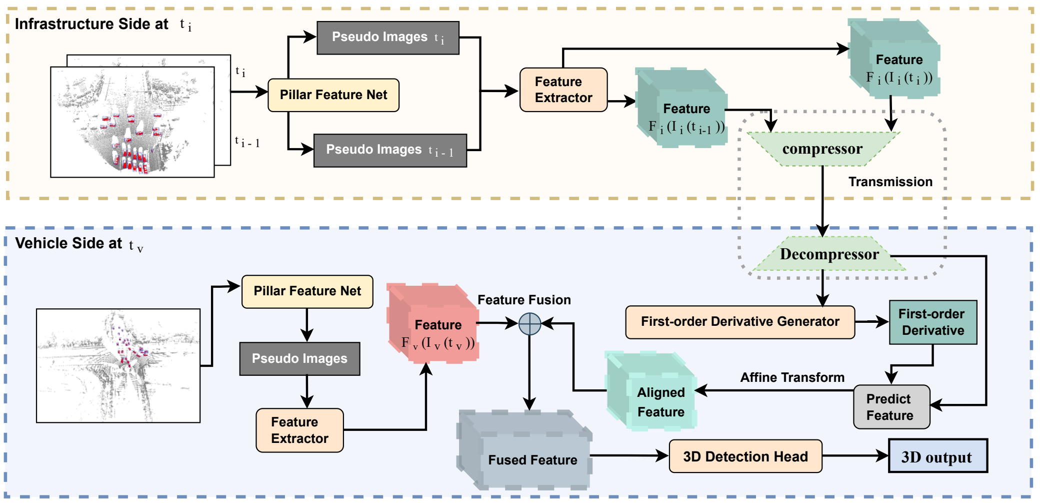


Figure 5: FFNet-V Overview. FFNet-V generates the feature flow and predicts the future feature with the compressed features received from the infrastructure. Calculating feature flow on vehicle uses compressed features as inputs, giving us less valuable information than using all raw point clouds as inputs.

infrastructure to generate the feature flow on ego vehicle. We first concatenates the two received features and feeds them into a first-order derivative generator to generate the estimated first-order derivative of the feature flow $\widetilde{F}_i'(t_i)$. Then we predict the future feature as Eq. 4. Apart from that, this FFNet-V shares the same architecture and training configuration as FFNet. We provide the FFNet-V implementation in Fig. 5. To keep the exact transmission cost as FFNet, we also train another FFNet-V by compressing the feature from (384, 288, 288) to (384/16, 288/8, 288/8). This FFNet-V with the exact transmission cost is called FFNet-V (Same-TC). We evaluate the FFNet-V and FFNet-V (Same-TC) in different latency like 100$ms$, 300$ms$, and 500$ms$. The experiment results are shown in Tab. 3.

In Tab. 3, both the FFNet-V and FFNet-V (Same-TC) compensate much less performance drop compared with FFNet in different latency, and this disadvantage becomes more pronounced over latency increases. Significantly FFNet outperforms FFNet-V (Same-TC) by more than 3% mAP@BEV (IoU=0.5) in 300$ms$ latency with the exact transmission cost. The results show that extracting feature flow on infrastructure can improve the VIC3D detection performance better than extracting feature flow on ego vehicle. In addition, FFNet needs much fewer ego-vehicle computing resources, and the computing cost complexity (CCC) is only O(N) because the feature flow is generated on infrastructure. Conversely, FFNet-V consumes computation resources up to O(N) to extract flow from past features. So the FFNet is more computation-friendly to ego vehicle. Moreover, extracting feature flow on ego vehicle requires much more storage because the feature flow extraction depends on the past frames that the ego vehicle received. At the same time, FFNet-V relies heavily on past consecutive frames, so dropped frames will seriously affect the execution and performance. So the FFNet is more storage-friendly to the ego vehicle and more robust to the frame dropping.