FSMDet: Vision-guided feature diffusion for fully sparse 3D detector

Tianran Liu[1], Morteza Mousa Pasandi[1], and Robert Laganiere[1]

VIVA Lab, Department of EECS, University of Ottawa, Ontario, Canada
{tliu157, mmous029, laganier}@uottawa.ca

Abstract. Fully sparse 3D detection has attracted an increasing interest in the recent years. However, the sparsity of the features in these frameworks challenges the generation of proposals because of the limited diffusion process. In addition, the quest for efficiency has led to only few work on vision-assisted fully sparse models. In this paper, we propose FSMDet (Fully Sparse Multi-modal Detection), which use visual information to guide the LiDAR feature diffusion process while still maintaining the efficiency of the pipeline. Specifically, most of fully sparse works focus on complex customized center fusion diffusion/regression operators. However, we observed that if the adequate object completion is performed, even the simplest interpolation operator leads to satisfactory results. Inspired by this observation, we split the vision-guided diffusion process into two modules: a Shape Recover Layer (SRLayer) and a Self Diffusion Layer (SDLayer). The former uses RGB information to recover the shape of the visible part of an object, and the latter uses a visual prior to further spread the features to the center region. Experiments demonstrate that our approach successfully improves the performance of previous fully sparse models that use LiDAR only and reaches SOTA performance in multimodal models. At the same time, thanks to the sparse architecture, our method can be up to 5 times more efficient than previous SOTA methods in the inference process.

Keywords: RGB-LiDAR Fusion · Fully sparse 3D detection · Shape recovery

1 Introduction

The 3D detection of objects using LiDAR has made great progress in the past few years. However, the vast majority of dense detector, whether they be anchor-based [25,4,41,24,15,21,16,2] or anchor-free [42,36] works on dense Bird's Eye View (BEV) representations which are costly to process. For instance, for every doubling of the detection range, the computational cost will increase by 2-4 times[7]. To overcome this issue, fully sparse detectors have witnessed an increased interest in recent years. Different from point-based methods[26,23,22,27], which perform time-consuming point neighborhood aggregation, fully sparse models tend to introduce a voxel or a point-wise segmentation to significantly reduce the computation cost early in the network.

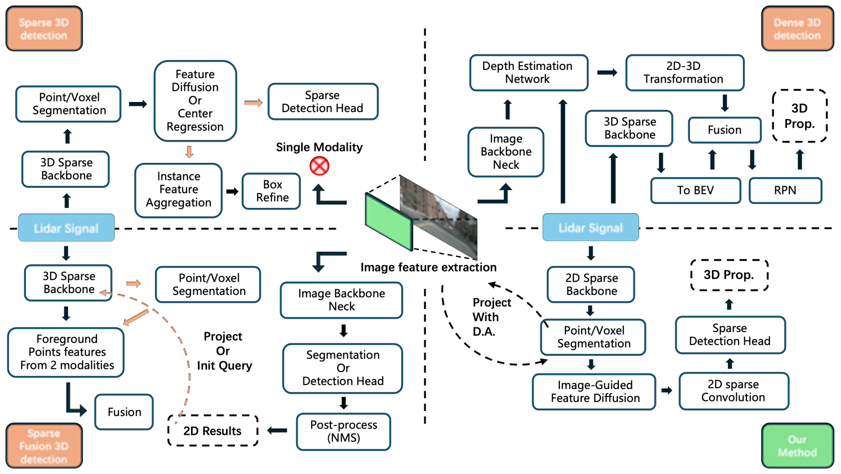

Fig. 1: A Vision-Centric view for LiDAR-RGB fusion in a fully sparse framework. Our proposed FSMDet integrates image features without any parameters for 2D tasks. The feature diffusion is guided by RGB features. Prop. and D.A. stand for proposals and deformable attention. The orange arrows represent the optional steps for models in the same categories.

However, sparse models come also with their issues. Two main questions in fully sparse detectors are center feature missing and image signal fusion. Specifically, for the detectors that adopt a dense detection head on the BEV map, the normal pipeline first feeds the voxelized lidar signal to backbone, which always consists of a stack of submanifold sparse convolution layers[10]. However, this step won't change the occupancy of voxels. After the backbone, the z-axis(height) will be flattened to obtain the BEV map, which will still be sparse. Normally, the 2D CNN will process the BEV maps to diffuse the features to the empty voxels and then the detection head can generate the proposals for every non-empty voxel. The feature located in the voxel at the object's center will be the main candidate to regress the predicted box of the object. However, this method won't solve the issues created by missing voxel center feature in a fully sparse pipeline. Many fully sparse pipelines propose modules that vote the center of objects directly[7] or use the voted center as virtual voxels[8] to further aggregate features. However, this is a complex strategy that requires several hand-craft parameter settings which makes the model far from satisfactory in terms of deployment and speed of inference. It is worth noting that the recently proposed SAFDNet[37] aims at achieving to achieve this goal through a foreground-only feature diffusion approach, but this parameter-free operation makes it difficult to ensure not only the accuracy of the diffusion but also the quality of the diffused features.

In this work, we demonstrate that vision information can be used to guide the feature diffusion in a 3D detection framework. Only a few works[5,33,18] tried to introduce the RGB features into a fully sparse pipeline. However, most of them need a 2D segmentation or detection head to localize the objects from the image signal. These modules which include RPN and NMS operations limit the overall inference speed of the 3D detection framework. In other words, 2D detection is only intended to be used as an assist to the subsequent 3D detection results. Such a simple merged network tends to introduce a large amount of parameter redundancy. This inefficiency in RGB fusion approaches in fully sparse networks led us to revisit how to introduce RGB information while maintaining architectural sparsity. This paper introduces the FSMDet network (Fully Sparse Multi-modal Detection), a straightforward and efficient multi-modal 3D detector, that only keeps the necessary parameters for the image-branch and retains the sparsity of the structure.

As shown in Fig 1, our proposed method is as follows: the voxelized LiDAR features interact with RGB features through deformable attention operations to select the foreground RGB features. The foreground voxel features are diffused in 2 ways: using the shape-recovery layer (SRLayer) and the self-diffusion layer (SFLayer). Specifically, considering that the diffusion process changes the occupancy, we believe that the process of an ideal feature diffusion should essentially be like shape completion. From an idealized test, we experimentally proved that for a fully sparse model, even if there is no specific design for the center feature diffusion, we can still get nearly 100% accuracy with the completed objects. This conclusion can also be extended to most point-based methods. However, since the full shape of objects is not always available in the image, it is not a trivial task to make a network predicting the full shape of objects. Therefore, we further tested the visible part completion with color information. The result shows that this is even better for small objects with almost the same performance in other categories compared with the full-shape competition.

The experiment shows that a shape recovery operation at the early stage of the network is essential for accuracy. Inspired by this observation, the SRLayer is used at an early stage of the network to recover the shape of the visible part with features from the camera and the LiDAR modalities. The self-diffusion layer (SDLayer) is performed before the sparse detection head, which expands the feature at the boundary of objects to the center, thus solving the center feature estimation problem mentioned before.

Our proposed FSMDet network efficiently and effectively introduces visual information into a fully sparse detector: on a desktop GPU, without any CUDA kernel or deployment optimization, our method achieve a competitive accuracy comparing with SOTA fusion-based solutions with up to 5.3x speedup.

2 Related Work

Fully sparse 3D detection. A premise that has sometimes been overlooked in past 3D detection methods[4,24,19] is the different importance of differentiating foreground objects from the background which significant impact computing resources. FSD [6] first proposed a solution with a segmentation head to distinguish objects from background signals and a voting network to regress the center of objects. Subsequently FSDv2[8] improves the performance by introducing virtual voxels and a multi-stage pipeline. Later, the FSD++[9] utilized multi-frame information to improve the segmentation process. Considering the complexity of the pipeline, VoxelNeXt [3] adopt a relatively simple structure with extra down-sampling steps, which get better results with lower latency. The SAFDNet[37] proposed another strategy that used a 2D sparse convolution layer named AFD to diffuse the foreground lidar signal. Although most of the works achieved the SOTA performance, none of the mentioned works introduced multi-modal signal analysis into the fully sparse 3D detection pipeline.

Object Completion in 3D detection. The concept of object completion starts from lidar-based 3D detection: With a sub-network recovering the missing signal of distant or occluded objects, the performance of 3D detection can be boosted. Specifically, BtcDet[34] used cylinder coordination to voxelise the space and then try to recover the missing signal by predicting the occupancy of voxels. SPG[35] tries to use an unsupervised method to expand the foreground voxels. Both PG-RCNN[14] and PC-RGNN[38] tried to densify the foreground region after proposing the region in the first stage. Sparse2Dense [30] proposed a new way to complete the object in the hidden space by an explicit optimization. This same idea is behind many lidar-RGB fusion solutions. SFDNet[32] and VirConv[31] introduce dense pseudo points to complete the objects and the latter further proposed different modules to remove possible noise. Distinct from these 2 works, WYSIWYD[20] introduces a mesh deformation method that only completes the depth of foreground parts visible in the RGB image.

3 FSMDet

3.1 Preliminary

Previous works in fully sparse detection [8,37,7] attribute its performance flaw to center feature aggregation, and then proposed different methods to improve the performance of this module. In this section, we introduce a series of ideal experiments to show that another important factor that affects the performance of a sparse 3D detector is the density of the foreground objects point cloud representation. In this first experiment, our objective is to test the performance of sparse models with plain center feature aggregation modules under different lidar inputs. Note that when choosing the models to be tested, it is important to select ones that do not change the center voxels' occupancy. This way, the change in performance can be attributed to the different lidar inputs.

With a 3D ground truth box B_i of object i, we can obtain the lidar point set in this box, denoted by L_i. Using the full shape(FS) object point completion methods described in [34,20], we can obtain a densified object, denoted by $\widetilde{L_i}$,

Models	Comp	Car 3D AP_{R40}			Car BEV AP_{R40}			Ped 3D AP_{R40}			Ped BEV AP_{R40}		
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
PointRCNN	VP	99.9	99.6	97.2	99.9	99.6	97.2	85.3	77.9	86.2	68.1	80.9	71.2
	FS	97.2	97.2	97.1	97.2	97.2	97.1	77.8	72.8	70.3	80.3	74.8	71.1
IA-SSD	VP	99.9	99.7	99.4	99.9	99.3	99.3	75.5	71.3	66.4	77.5	75.1	70.3
	FS	99.7	99.4	99.4	99.6	99.2	99.4	71.9	69.7	86.2	74.8	74.3	72.8
FSD-V1	VP	99.9	99.7	99.7	99.9	99.7	99.7	87.2	80.1	72.1	89.9	82.7	73.6
	FS	99.9	99.9	99.9	99.9	99.9	99.9	81.7	77.9	71.1	83.3	77.1	72.9

Table 1: The performance of different sparse models with completed objects on the KITTI validation set. FS and VP stand for full shape and Visible Part completion respectively.

as shown in Fig 3. Our experiment, under ideal condition, consists then in completing the full shape of all ground-truth foreground objects in a scene. For this experiments, we use 3 classic models: IA-SSD[39], Point-RCNN[26], and FSD[7]. As shown in the FS line of Table 1, even for these models without a modern center feature aggregation module, good performance can be obtained if the objects are completed. This suggests that even the most basic center regression method is sufficient for well-completed objects to give us good results in 3D detection.

However, although full-shape completion can lead to a good results, it is still a non-trivial task to recover the full shape either from the image or the lidar signal. Therefore, we further use the obtained $\widetilde{L_i}$ to generate the visible part of objects, denoted by $\hat{L}_i$. Specifically, for every single L_i in the dataset, a unique 2D instance I_i on the image can be identified. First, we adopted a surface reconstruction from the Possion[13] method, and then placed the obtained hull(reconstructed surface), denoted by $\mathcal{M}_i$ in 3D space. Based on the known camera intrinsic matrix, extrinsic matrix, and the pixel 2D coordinates of pixels in the region I_i, a ray casting model can be established. Note that since the rays from the pixel on the boundary sometimes miss the object, a volume expansion coefficient δ is used here to make sure there is always a bijection from the pixel set to the depth set. This process is illustrated in Figure 2.

With the equation of line T_c and $\mathcal{M}_i$, the distance traveled by a ray of light from point c when it arrives on the mesh can be expressed as $|\delta\mathcal{M}_i \cap T_c - c_i|$. Considering the slope of T_C, the depth of the point on the mesh hit by the light to the camera plane is d_i. Here $\cdot$ refers to scalar multiplication.

$$d_c = |\delta\mathcal{M}_i \cap T_c - c_i| \cdot |\boldsymbol{x_c}| \tag{1}$$

After estimating the depth of every pixel in the region I_i, we get a dense visible surface of the object, denoted by $\hat{L}_i$, as shown in the left most sub-figure of Fig 3. Replacing the original $\widetilde{L_i}$ with $\hat{L}_i$, we repeated the same experiment on the selected models in VP lines of Table 1, show that even with only visible part completion, the upper boundary of these models is still high enough. A

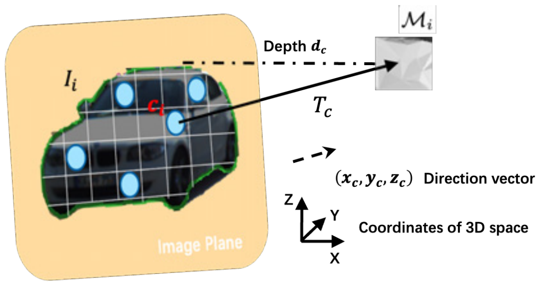

Fig. 2: The ray casting model for visible part ground truth generation. All calculations are performed in LiDAR(3D space) coordination. Here c_i stand for the location of pixel c in 3D space.

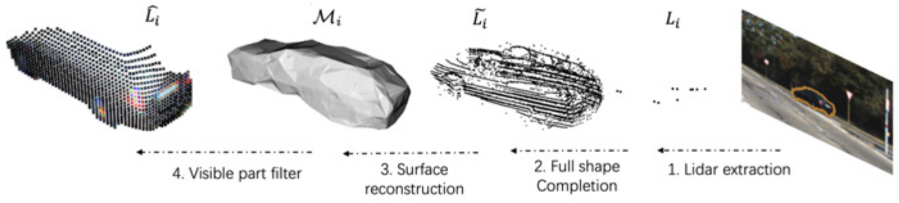

Fig. 3: From sparse lidar signal to visible part ground truth: we first obtain the full shape objects and then reconstruct the surface. With a ray-cast model, we can paint the color information onto the object's surface and filter out the non-visible part.

counter-intuitive fact that also needs to be mentioned is that VP is even more suitable for small objects such as pedestrians.

During the training, the completion of objects causes a change of occupancy, or in other words, it diffuses the point features. This experiment shows that the detection results are satisfactory when the object shapes are adequately completed, even with a vanilla design for the center regression or center feature aggregation modules. This inspired us to split the feature diffusion into two steps: the shape recovery diffusion and the diffusion of features to the object center. Since only the visible part of objects is available on the image, here we use VP to supervise the first-step diffusion.

3.2 Overall structure

In Section 3.1, we experimentally demonstrated that the visible part completion of the foreground object in the LiDAR signal improve the detection results despite the diffculty of center prediction. Based on this result, we introduce the

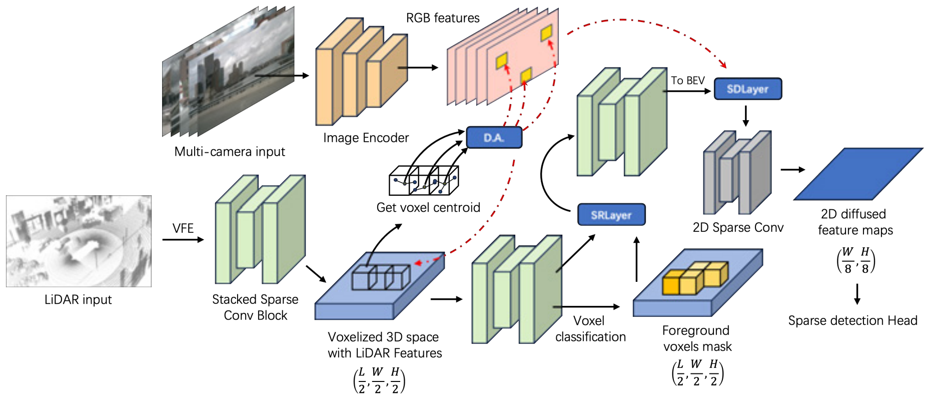
RGB features
Multi-camera input
Image Encoder
To BEV
SDLayer
D.A.
Get voxel centroid
SRLayer
2D Sparse Conv
2D diffused feature maps
$\left(\frac{W}{8},\frac{H}{8}\right)$
Sparse detection Head
LiDAR input
VFE
Stacked Sparse Conv Block
Voxelized 3D space with LiDAR Features
$\left(\frac{L}{2},\frac{W}{2},\frac{H}{2}\right)$
Voxel classification
Foreground voxels mask
$\left(\frac{L}{2},\frac{W}{2},\frac{H}{2}\right)$

Fig. 4: Proposed pipeline: We complete the diffusion of features in two steps, the SRLayer uses RGB features to recover the visible part of the contours of objects while the SDLayer diffuses the features to the center of objects. D.A. stand for the deformable attention. The operation related to color features is marked in red.

shape recovery module, which diffuses the features to occupy near voxels, at an early layer of the network. Our experiments will also demonstrate that it's beneficial to do this step at an early stage.

Specifically, as shown in Fig 4, we use SRLayer (Shape Recover Layer) and SDLayer (Self Diffusion Layer) to do the two steps mentioned respectively. After the voxelization of the space, Submanifold sparse convolution[11] with residual design was used as the basic block to process the LiDAR data. As in work [37], we call these stacked sparse convolution block **SRB**(Sparse Residual Block). We choose $\frac{1}{4}$ of the original size to perform shape recovery and color features aggregation as a good balance between efficiency and performance. With the coordinates of points centroid in the different voxels and the calibration information, we fuse the voxel and image features with a deformable attention operator and replace the original feature. These augmented features will be further process with SRBs, and then a voxel classification head will be applied to get the foreground parts of the scene. The fused features and the foreground masks are used as the input of our SRLayer. SRLayers use 2D VP object completion as the supervision signal.

The output of SRLayers will go through the SRBs for further downsampling. Finally, the BEV transformation squeezes the feature map to get a 2D representation of the space, the SDLayer will be used to deliver the feature to the center. The output size of the proposed cross-modality backbone will be $\frac{1}{8}$ of the width and length of the original input.

3.3 Proposed Method

Deformable attention for feature fusion. Let $V = \{V_i, V_{f_i}, c_i\}_{i=1}^{|V|}$ represent all of the non-empty voxels in a sparse feature map, V_i is the indices of the voxel i, V_{f_j} stand for the features in the voxel and c_i is the centroid of the LiDAR points in this voxel. With the camera intrinsic and extrinsic matrix, the 3D-2D transformation can be denoted as $\mathcal{P}$, to obtain the 2D-pixel coordinates $\mathbf{p}_i$, as shown in equation 2.

$$\mathbf{p}_i = \mathcal{P}(c_i) \tag{2}$$

Normally, we will interpolate the image features to the original size and acquire the image feature for points in 2D coordinates. However, as pointed out in previous lidar segmentation work[40], the calibration of different modalities is not always reliable. Therefore the correspondency between semantic information in down-sampled image feature maps and voxel centroid point is not guaranteed. For this reason, we introduce deformable attention in this step to allow the network to freely select features near $\mathbf{p}_i$, following the design of LoGoNet[17] with a minor modification.

Specifically, V_{f_i} will be used as query of the attention calculation. The image features sampled from the pixels around $\mathbf{p}_i$, denoted by $\hat{R}_i$, will be used as Key and Value. The image features located at $\mathbf{p}_i$ and all of its 3-order neighbour will be noted as $R_{\mathbf{p}_i}$ and $R^3_{\mathbf{p}_i}$, with the image map itself denoted by R. With M attention head and K sample points, we can obtain $\hat{R}_i$ from Equation 3.

$$\begin{gathered}\Delta\mathbf{p}_{mki} = MLP(MLP(R^3_{\mathbf{p}_i}) \cdot V_{f_i}) \\ R_{mi} = \mathcal{G}(R, \mathbf{p}_i + \Delta\mathbf{p}_{mki}) \\ \hat{R}_i = \sum_{m=1}^{M} W_m \left[\sum_{k=1}^{K} A_{mik} \cdot (W'_m R_{mi})\right]\end{gathered} \tag{3}$$

The $\mathcal{G}(a, b)$ represents the operation of obtaining the features of a at location b. The W'_m, W_m stands for the trainable weight, and A_{mik} stands for attention weight. In the original design, the $\Delta\mathbf{p}_{mki}$ was only decided by the LiDAR feature V_{f_i}, however, here we add an interaction between V_{f_i} and all features around $R_{\mathbf{p}_i}$, which will help to identify the proper features to be aggregated.

Shape Recover Layer. The augmented features will replace the original one and is sent to the next SRBs. Note that at this stage, we keep the original feature map size. The obtained map is then sent to a classification head to split the foreground voxels from the background information. Considering the large space difference between the various classes of objects, we set a category number $\mathcal{S}$ for the classification. Using $\mathcal{M}$ to represent the index of foreground voxels, the $\mathcal{M}$ and processed voxels $V' = \left\{V_i, V'_{f_i}, c_i\right\}_{i=1}^{|V|}$ will be sent to the SRLayer.

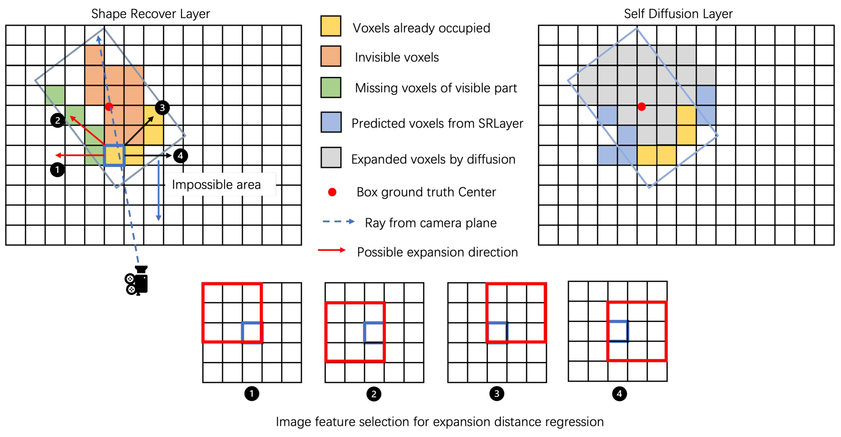
Shape Recover Layer
Self Diffusion Layer
Voxels already occupied
Invisible voxels
Missing voxels of visible part
Predicted voxels from SRLayer
Expanded voxels by diffusion
Box ground truth Center
Ray from camera plane
Possible expansion direction
Impossible area
1
2
3
4
Image feature selection for expansion distance regression

Fig. 5: Illustration of Shape Recover Layer and Self Diffusion Layer. We use the sparse signal to recover the shape of objects. After Shape recovery, we expand the boundary voxels along the ray direction, the center point will be naturally located in this area.

With the foreground voxels that are denoted by $V'_o = \left\{V_i, V'_{f_i}, c_i | V_i \in \mathcal{M}\right\}_{i=1}^{|\mathbf{M}|}$, we first need to calculate the possible direction of diffusion. A possible direction should ensure that the voxel in that direction is visible on the image and secondly that the next two voxels in that direction are not occupied. As shown in the left part of Fig 5, all voxels in V'_o will have 4 candidate directions to diffuse, but here only 2 of them are possible for diffusion (marked with red). Specifically, for a specific voxel in V'_o, denoted by $\left\{V_t, V'_{f_t}, c_t\right\}$, which can be projected to a pixel located at $\mathbf{p}_t$, we use $V_\varnothing$ to denote all invisible voxels. The possible diffusion direction J_t can be described in Equation 4. Here V_j is the nearest non-empty voxels on direction J_t.

$$J_t = \{J_{ti} | J_{ti} \cap V_\varnothing, \|V_t - V_j\| < 2\}_{i=1}^{4} \tag{4}$$

For every J_{ti}, we use V'_{f_t} and the direction-specific image features (as shown in the bottom of Figure 5), denoted by $R^i_{\mathbf{p}_t}$, to regress the expand distance d_t. For the empty voxel in range d_t alone the direction J_t, denoted by $\left\{V_{t+d}, V'_{f_{t+d}}, c_{t+d}\right\}$, the $V'_{f_{t+d}}$ will be obtain by equation 5.

$$V'_{f_{t+d}} = MLP\left(\bigcup_{\mathbf{N}(p_{t+d},d)} R^i_{\mathbf{p}_{t+d}}\right) \tag{5}$$

Here $\mathbf{N}(x, l)$ and $\bigcup$ stand for the l-order neighborhood of pixel x and the concatenation operation of the neighboring features. With the new occupied voxels, the V' will be sent to the next SRB to process the features further.

The Shape Recover Layer needs supervision from the generated VP ground truth. Considering the computational complexity, we only apply the supervision on the BEV plane. Specifically, after voxelizing the VP ground truth, we can get the ground truth for the shape recovery, denoted by V_{gt}. Using $\hat{V}$ to represent the score of predicted recovery voxels, the loss can be written as Equation 6.

$$\mathcal{L} = \sum_{i=1}^{\mathbf{S}} \alpha_j \sum_{j=1}^{3} focal\_loss(\hat{V_{ij}}, V_{gt}) \tag{6}$$

Note that the boundary voxels only make up a small portion. Here we use a multi-class focal loss to calculate if the predicted occupied voxel cannot cover the boundary ground truth. If it is located in the 2D box, it won't be treated as negative samples. Here we further split the voxel predicted for different size objects into 3 categories: the predicted voxels located at the boundary correctly, the predicted voxels not situated at the boundary but in the box, and the others. Specific α_j will be used to scale these different cases here.

Self Diffusion Layer. Before being sent to the detection head, the sparse signal needs to be further diffused to the center of objects. Note that since we have identified the invisible voxels $V_{\varnothing}$, we can directly diffuse the features along the ray across the occupied voxels. As we have shown in the right part of Figure 5, in most cases, the center can be naturally covered. Considering the size of different objects, the scope parameter σ_0 of this diffusion depends on the result of the classification head. Following SAFDNet[37], we also adopt an adaptive method in which for larger objects, the diffusion distance alone ray casting will be larger than that of small objects. However, when the objects are occluded, the diffusion along the ray-casting direction will miss the center. Therefore, considering the size of voxels here, we also set a parameter σ_1 to control the diffusion of features perpendicular to the direction of the ray casting.

4 Experiments

4.1 Dataset and Metrics

We conducted experiments on the nuScenes [1] dataset for a comparison. The nuScenes dataset was collected with 6 cameras, a radar sensor, and a 32-beam LiDAR sensor under different weather and illumination conditions. The model proposed was evaluated by the official evaluation script with NDS and MAP as index.

4.2 Implementation Details

We used the OpenPCDet[28] as the framework to implement the mentioned design. All experiments were performed on RTX3090, we used 8 as batch size on 4 GPUs for training and 1 batch size for inference speed test. Training in all experiments lasted for 20 epochs. During training, in addition to random rotation and translation of the point cloud signals and random cropping and normalization of the image signals, we change the GT-sampling commonly used in the lidar-based model to the multimodal GT-sampling proposed in PointAugmenting[29]. We use ResNet[12] pre-trained on NuImage as the image backbone. The volume expansion parameter δ was set to 1.15 to ensure all rays from every pixel wouldn't miss the object. d in Equation 5 was set to 2. Considering the size of different objects occupied, we set $\sigma_0 = 6$ and $\sigma_1 = 4$. We set the $\alpha_0 = \alpha_1 = 0.5$, $\alpha_2 = 1$ to scale the loss mentioned in Equation 6.

4.3 Main results

Methods	mAP↑	NDS↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
FSDv2	65.4	70.8	0.270	0.248	0.272	0.207	0.187
SAFDNet	68.3	72.3	0.251	0.242	0.311	0.258	0.127
SparseFusion	71.0	73.1	0.277	0.247	0.270	0.253	0.188
MSMDFusion	69.2	72.0	0.283	0.254	0.282	0.252	0.185
BEVFusion	67.3	70.7	0.286	0.255	0.313	0.251	0.186
FSMDet	70.0	72.2	0.285	0.251	0.296	0.254	0.179

Table 2: Performance Comparsion with SOTA models on **nuScences validation set**. The best results are shown in bold. ↑ higher is better, ↓ lower is better.

We compared the inference speed with recent SOTA models. Since we did not need any parameters for tasks on the image plane, our proposed method takes only 25%(as a minimum) of the time the SOTA solutions use to take while maintaining a decent detection performance. As shown in Table 3, our proposed method obtained 70.1 mAP and 72.1 NDS in nuScenes Dataset. This performance is only 2.4% to 3.8% less than the SOTA but we only used 18% to 47% of the inference time that is needed for SOTA models.

Model	Modality	Sparsity	Image Result	FPS	mAP	NDS
SAFDNet	LiDAR	✓	N/A	15.7	68.3	72.3
SparseFusion	LiDAR+RGB	✓	✓	5.3	71.0	73.1
MSMDFusion	LiDAR+RGB	✕	✕	2.1	71.4	73.9
BEVFusion	LiDAR+RGB	✕	✕	8.1	67.3	70.7
FSMDet	LiDAR+RGB	✓	✕	11.2	70.0	72.2

Table 3: Efficiency comparison, our proposed model achieves a balance point between inference speed and performance.

4.4 Ablation Study

In Table 4, we break down the performance boost in the proposed model. Here the D.A Color and Proj Color stand for the two different methods to introduce RGB color features: the deformable attention or direct projection. As shown in the table, the parameter-based solution benefits from the contribution of these models. In experiments on Table 4, if we do not apply the SDLayer, the ADF in SAFDNet[37] is used.

Models	D.A Color	Proj Color	SRLayer	SDLayer	mAP	NDS
FSMDet	✓	-	-	-	67.9	70.4
FSMDet	-	✓	-	-	67.6	70.8
FSMDet	✓	-	✓	-	69.8	71.5
FSMDet	✓	-	✓	✓	70.0	72.2

Table 4: The verification of the effect of each module on the model on the final accuracy

Here in Table 5, we compare the effect of different positions of the SRLayer on accuracy. If we insert the shape recovery operation too early, the shallow representation of LiDAR features can barely help the expansion regress. On the other hand, if the SRLayer is too close to the SDLayer, the feature of the newly occupied layer also cannot aggregate the nearby features very well.

Modules	Block-1	Block-2	Block-3	Block-4	mAP	NDS
SRLayer	✓	-	-	-	61.2	63.6
SRLayer	-	✓	-	-	70.0	72.2
SRLayer	-	-	✓	-	68.7	70.9
SRLayer	-	-	-	✓	63.1	67.2

Table 5: Different stages to insert the Shape Recover Layer

5 Conclusion

In this work, we proposed a cross-modality method for the RGB-LiDAR fusion in a fully sparse framework. Starting from the idealized experiment which reveals that the core of center regression for detection models should be the visible part completion, we designed the SRLayer to achieve this step with the guidance from the image features. This way, the diffusion alone of the ray from the image plane can naturally cover the center of objects in most cases. As the experiments prove the potential of our method, we hope the FSMDet can promote the research in data fusion for fully sparse models.

References

1. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving (2020), https://arxiv.org/abs/1903.11027 10
2. Chen, Y., Li, Y., Zhang, X., Sun, J., Jia, J.: Focal sparse convolutional networks for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5428–5437 (2022) 1
3. Chen, Y., Liu, J., Zhang, X., Qi, X., Jia, J.: Voxelnext: Fully sparse voxelnet for 3d object detection and tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21674–21683 (2023) 4
4. Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., Li, H.: Voxel r-cnn: Towards high performance voxel-based 3d object detection. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 1201–1209 (2021) 1, 3
5. Erabati, G.K., Araujo, H.: Srfdet3d: Sparse region fusion based 3d object detection. Neurocomputing **593**, 127814 (2024). https://doi.org/https://doi.org/10.1016/j.neucom.2024.127814, https://www.sciencedirect.com/science/article/pii/S092523122400585X 3
6. Fan, L., Wang, F., Wang, N., Zhang, Z.X.: Fully sparse 3d object detection. Advances in Neural Information Processing Systems **35**, 351–363 (2022) 4
7. Fan, L., Wang, F., Wang, N., Zhang, Z.: Fully sparse 3d object detection (2022), https://arxiv.org/abs/2207.10035 1, 2, 4, 5
8. Fan, L., Wang, F., Wang, N., Zhang, Z.: Fsd v2: Improving fully sparse 3d object detection with virtual voxels. arXiv preprint arXiv:2308.03755 (2023) 2, 4
9. Fan, L., Yang, Y., Wang, F., Wang, N., Zhang, Z.: Super sparse 3d object detection. IEEE transactions on pattern analysis and machine intelligence **45**(10), 12490–12505 (2023) 4
10. Graham, B., Engelcke, M., van der Maaten, L.: 3d semantic segmentation with submanifold sparse convolutional networks. CVPR (2018) 2
11. Graham, B., van der Maaten, L.: Submanifold sparse convolutional networks (2017), https://arxiv.org/abs/1706.01307 7
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) 11
13. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proceedings of the fourth Eurographics symposium on Geometry processing. vol. 7 (2006) 5
14. Koo, I., Lee, I., Kim, S.H., Kim, H.S., Jeon, W.J., Kim, C.: Pg-rcnn: Semantic surface point generation for 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 18142–18151 (2023) 4
15. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12697–12705 (2019) 1
16. Li, J., Luo, C., Yang, X.: Pillarnext: Rethinking network designs for 3d object detection in lidar point clouds (2023), https://arxiv.org/abs/2305.04925 1
17. Li, X., Ma, T., Hou, Y., Shi, B., Yang, Y., Liu, Y., Wu, X., Chen, Q., Li, Y., Qiao, Y., He, L.: Logonet: Towards accurate 3d object detection with local-to-global cross-modal fusion (2023), https://arxiv.org/abs/2303.03595 8
18. Li, Y., Fan, L., Liu, Y., Huang, Z., Chen, Y., Wang, N., Zhang, Z.: Fully sparse fusion for 3d object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence (2024) 3

19. Liang, T., Xie, H., Yu, K., Xia, Z., Lin, Z., Wang, Y., Tang, T., Wang, B., Tang, Z.: Bevfusion: A simple and robust lidar-camera fusion framework. Advances in Neural Information Processing Systems **35**, 10421–10434 (2022) 3
20. Liu, T., Zhang, Z., Pasandi, M.M., Laganiere, R.: What you see is what you detect: Towards better object densification in 3d detection (2023), https://arxiv.org/abs/2310.17842 4
21. Mao, J., Xue, Y., Niu, M., Bai, H., Feng, J., Liang, X., Xu, H., Xu, C.: Voxel transformer for 3d object detection. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 3164–3173 (2021) 1
22. Qi, C.R., Litany, O., He, K., Guibas, L.J.: Deep hough voting for 3d object detection in point clouds. In: Proceedings of the IEEE International Conference on Computer Vision (2019) 1
23. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space (2017), https://arxiv.org/abs/1706.02413 1
24. Shi, S., Guo, C., Jiang, L., Wang, Z., Shi, J., Wang, X., Li, H.: Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10529–10538 (2020) 1, 3
25. Shi, S., Jiang, L., Deng, J., Wang, Z., Guo, C., Shi, J., Wang, X., Li, H.: Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. International Journal of Computer Vision **131**(2), 531–551 (2023) 1
26. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud (2019), https://arxiv.org/abs/1812.04244 1, 5
27. Shi, W., Ragunathan, Rajkumar: Point-gnn: Graph neural network for 3d object detection in a point cloud (2020), https://arxiv.org/abs/2003.01251 1
28. Team, O.D.: Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet (2020) 10
29. Wang, C., Ma, C., Zhu, M., Yang, X.: Pointaugmenting: Cross-modal augmentation for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11794–11803 (June 2021) 11
30. Wang, T., Hu, X., Liu, Z., Fu, C.W.: Sparse2dense: Learning to densify 3d features for 3d object detection. Advances in Neural Information Processing Systems **35**, 38533–38545 (2022) 4
31. Wu, H., Wen, C., Shi, S., Wang, C.: Virtual sparse convolution for multimodal 3d object detection. In: CVPR (2023) 4
32. Wu, X., Peng, L., Yang, H., Xie, L., Huang, C., Deng, C., Liu, H., Cai, D.: Sparse fuse dense: Towards high quality 3d detection with depth completion. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 5418–5427 (2022) 4
33. Xie, Y., Xu, C., Rakotosaona, M.J., Rim, P., Tombari, F., Keutzer, K., Tomizuka, M., Zhan, W.: Sparsefusion: Fusing multi-modal sparse representations for multi-sensor 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 17591–17602 (2023) 3
34. Xu, Q., Zhong, Y., Neumann, U.: Behind the curtain: Learning occluded shapes for 3d object detection. arXiv preprint arXiv:2112.02205 (2021) 4
35. Xu, Q., Zhou, Y., Wang, W., Qi, C.R., Anguelov, D.: Spg: Unsupervised domain adaptation for 3d object detection via semantic point generation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15446–15456 (2021) 4

36. Yin, T., Zhou, X., Krahenbuhl, P.: Center-based 3d object detection and tracking. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11784–11793 (2021) 1
37. Zhang, G., Chen, J., Gao, G., Li, J., Liu, S., Hu, X.: SAFDNet: A simple and effective network for fully sparse 3d object detection (2024) 2, 4, 7, 10, 12
38. Zhang, Y., Huang, D., Wang, Y.: Pc-rgnn: Point cloud completion and graph neural network for 3d object detection. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 3430–3437 (2021) 4
39. Zhang, Y., Hu, Q., Xu, G., Ma, Y., Wan, J., Guo, Y.: Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds (2022), https://arxiv.org/abs/2203.11139 5
40. Zhang, Z., Zhang, Z., Yu, Q., Yi, R., Xie, Y., Ma, L.: Lidar-camera panoptic segmentation via geometry-consistent and semantic-aware alignment (2023), https://arxiv.org/abs/2308.01686 8
41. Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4490–4499 (2018) 1
42. Zhou, Z., Zhao, X., Wang, Y., Wang, P., Foroosh, H.: Centerformer: Center-based transformer for 3d object detection (2022), https://arxiv.org/abs/2209.05588 1